

Designing Human-Centered Autonomous Agents

Gregory Dorais

NASA Ames Research Center

David Kortenkamp

NASA Johnson Space Center

Outline

- Introduction
- Related research areas
- Designing a Human-Centered Autonomous system
- Requirements that HCA places on software systems
- Existing NASA HCA applications
- Summary

Introduction

- Human-centered automation (HCA)
 - maximizes goals of humans
 - supports full range of interactions
- Want to minimize the necessity for human interaction, but maximize the capability to interact
- Synergism between operators and the autonomous system
- Adjustable autonomy
 - allows autonomous systems to operate with dynamically varying levels of independence, intelligence and control

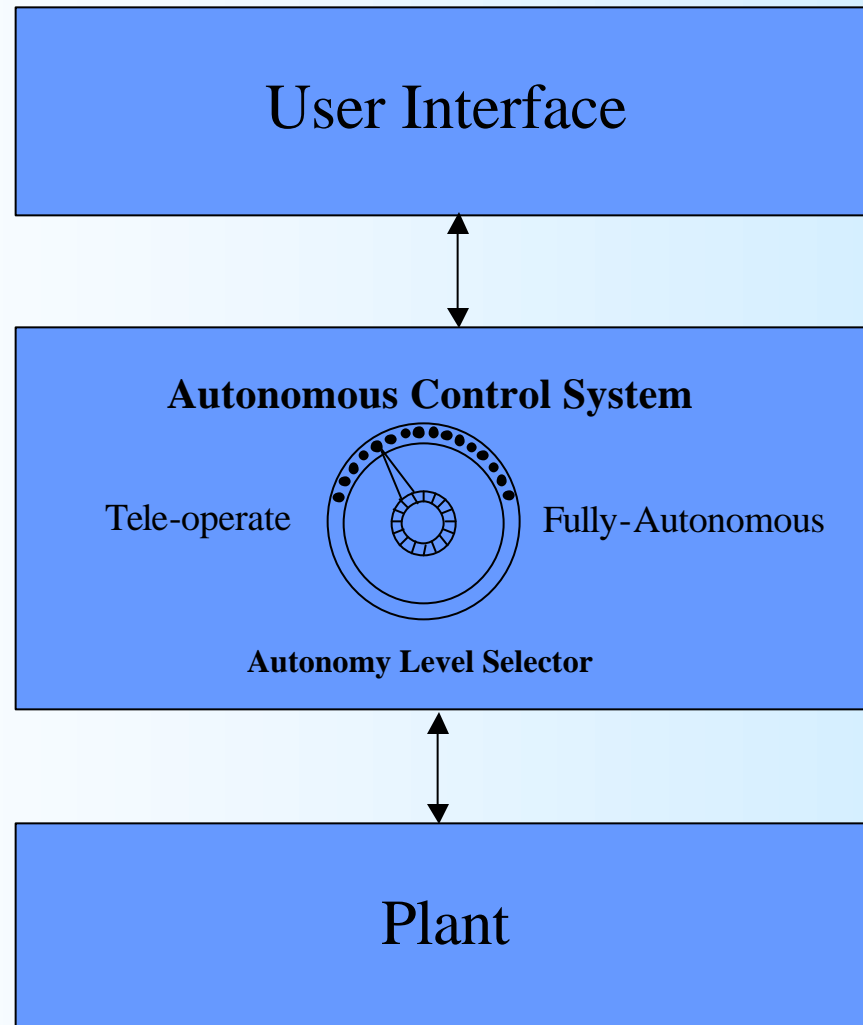
Adjustable Autonomy Systems

A control system that has the ability to:

- be completely in control
- supervise manual control
- be somewhere in between
- shift among these control extremes in a safe and efficient manner

is an *adjustable autonomy* system

Basic HCA Architecture



Adjustable Autonomy

- A system's adjustable autonomy can involve changes in:
 - The complexity of the commands it executes
 - The resources (including time) consumed by its operation
 - The circumstances under which it will either override or allow manual control
 - The circumstances under which it will request user information or control
 - The number of subsystems that are being controlled autonomously

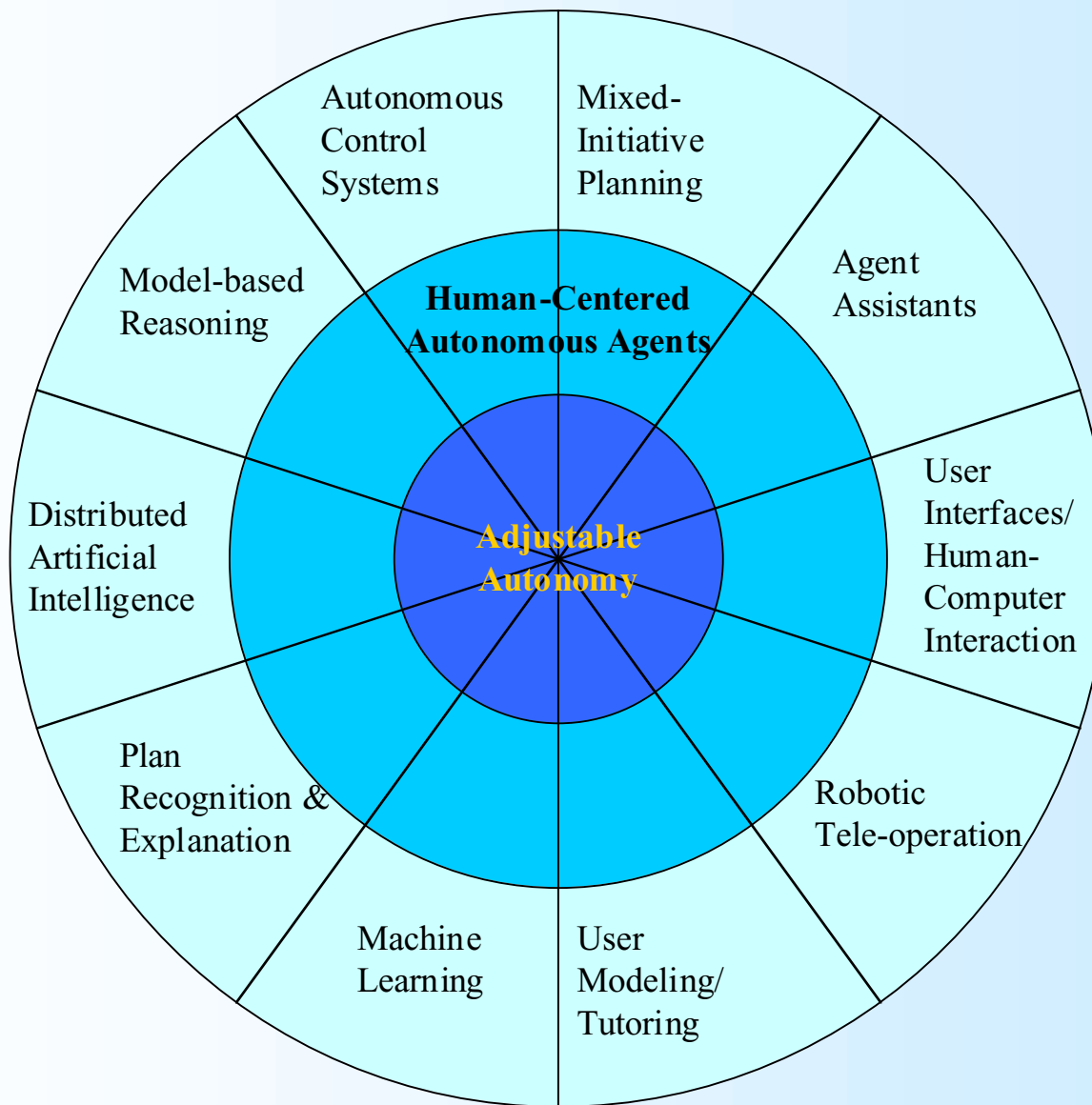
Motivations

- Complexity of commands
- Uncertainty/changing environment
- Safety/monitoring/politics
- Maintenance or calibration
- Training
- Flexibility
- Resource allocation

Benefits

- Partial autonomy where full autonomy not possible or desired
- Lower cost
 - difficult-to-automate parts of the system can be left for humans
- Safety and reliability
 - human experience brought to bear when needed
- Operator acceptance of autonomous systems

Related Work



Autonomous Control Systems

1 Brings to table:

- autonomous control
- integration of continuous and symbolic
- task contexts and off-nominal operation

1 Lacks:

- machinery for human interaction
- explanation facilities
- history of operations

1 Citations

- Muscettola, et al 1998
- Bonasso, et al 1997
- Musliner, et al 1995, Kuo, 1991

Mixed-Initiative Planning

- 1 Brings to table:
 - mechanisms for human involvement in plan generation
 - language for explaining choices to human
 - look-ahead search of options and consequences
- 1 Lacks
 - execution of plans
- 1 Citations
 - Ferguson, et al 1996
 - Burstein and McDermott, 1996
 - Pollack and Horty, 1999
 - Myers, 1996

Agent Assistants

- 1 Brings to table:
 - close human/computer interaction
 - studies of how humans and autonomous systems can work together safely
 - continuous sensor reasoning
- 1 Lacks
 - fully autonomous capabilities doesn't take charge
 - generalizable results domain specific
- 1 Citations
 - Chambers and Nagel, 1985
 - Decker and Lesser, 1995
 - Jorgensen, 1997

Human-Computer Interaction

1 Brings to table:

- careful studies of how to present information to humans
- careful studies of how to make commanding easier and less error-prone
- simulation tools for modeling work practices

1 Lacks

- autonomous control

1 Citations

- Roth, et al 1997
- Schreckenghost and Malin, 1991
- Gould, 1998
- Clancey, et al 1998
- Gertz, Stewart, and Khosla, 1993

Robotic Teleoperation

1 Brings to table:

- continuous control
- shared control (human and robot each controlling different things)
- studies in time lags between action and control
- virtual presence and user interfaces

1 Lacks

- autonomous control
- history of system decisions

1 Citations

- Sheridan, 1989, 1992
- Craig, 1947
- Hayati and Venkataraman, 1989
- Lee, 1993

User Modeling/Tutoring

- 1 Brings to table:
 - psychological studies about interacting with humans
 - mechanism for presenting appropriate information to human
 - internal models of expected human behavior
- 1 Lacks:
 - control mechanisms
- 1 Citations
 - Anderson, et al 1989
 - Horvitz, et al 1998

Machine Learning

1 Brings to table:

- automatic adjustments of control system
- learning user behaviors and desires
- adaptations to different humans and situations

1 Lacks:

- verifiable control strategies
- observability
- ability to change autonomy level

1 Citations

- Samuel, 1959
- Holland, 1992
- Mitchell, 1997
- Grefenstette, et al 1990

Plan Recognition and Explanation

- 1 Brings to table:
 - mechanisms for predicting user actions
 - mechanisms for explaining system activities
- 1 Lacks
 - autonomous control facilities
- 1 Citations
 - Huber, et al 1994
 - Kantz and Allen, 1986
 - Canamero, et al 1994
 - Lesh, et al 1999
 - Stein, 1988

Distributed AI

- 1 Brings to table:
 - reasoning about multiple (including human) agents
 - distributing tasks amongst agents
 - inter-agent communication
 - resource allocation
- 1 Lacks
 - focus on human to understand and command control system
- 1 Citations
 - Durfee, 1999
 - Tambe, et al 1999
 - Weiss (ed), 2000

Model-based Reasoning

1 Brings to table:

- qualitative reasoning that is intuitive for humans
- fault diagnosis and recovery
- explanation

1 Lacks

- autonomous control in real-time and continuous domains
- human interface and input

1 Citations

- Williams, 1996
- Kuipers, 1994
- Jonsson, et al 2000

Adjustable Autonomy

1 Lacks

- full spectrum of control
- verification
- understanding

1 Citations

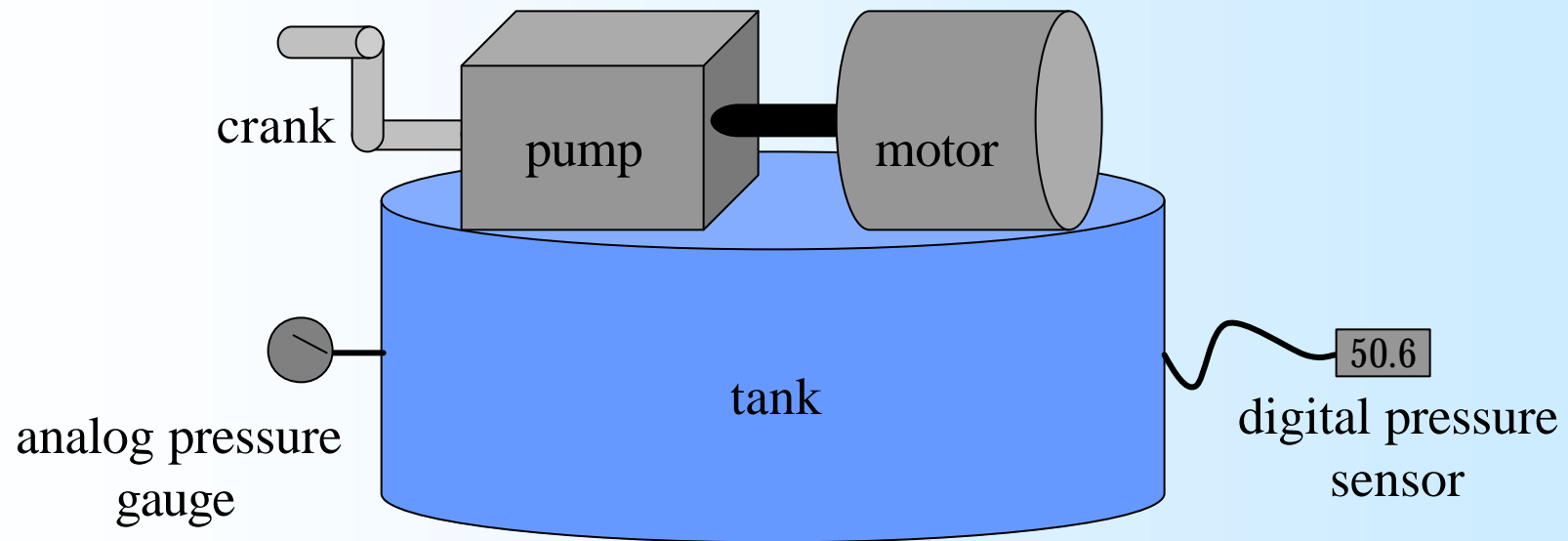
- Barber, et al 2000
- Bonasso, et al 1997
- Dorais, et al 1998
- Kortenkamp, et al 2000
- Musliner and Krebsbach, 1999
- Thurman, et al 1997

Design of AA systems

- Strive for full autonomy where possible.
- Build in appropriate sensing, even for situations in which the human is in control
- Build in capabilities that enable multiple methods for humans and system to achieve goals
- Plan for changes in autonomy as much as possible
 - system-planned changes in autonomy
 - system-initiated changes in autonomy
 - human-initiated changes in autonomy

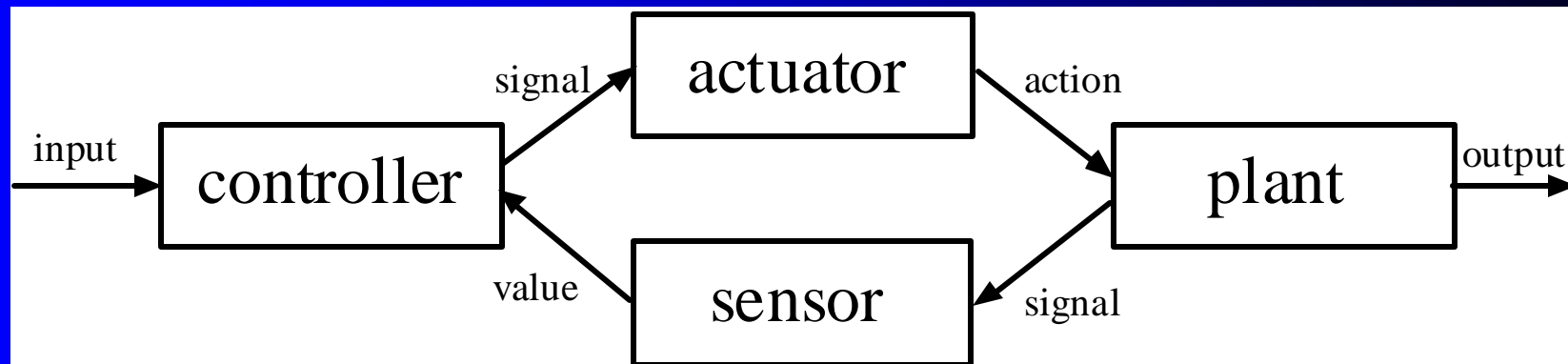
A simple example

- Simple example that will be discussed throughout this tutorial



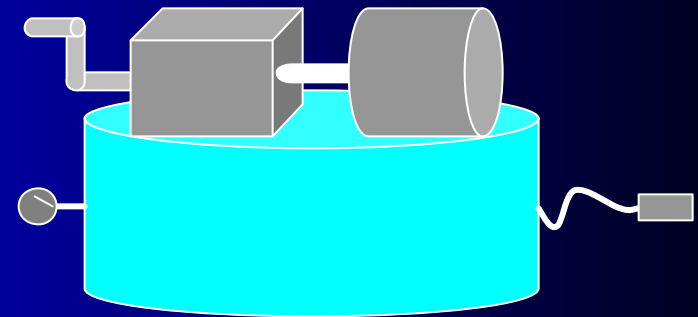
Generic control system

- Closed-loop control
 - Sensed output of plant (known via sensors) is compared to desired state of plant (input) and control of the plant through actuators is adjusted by the controller

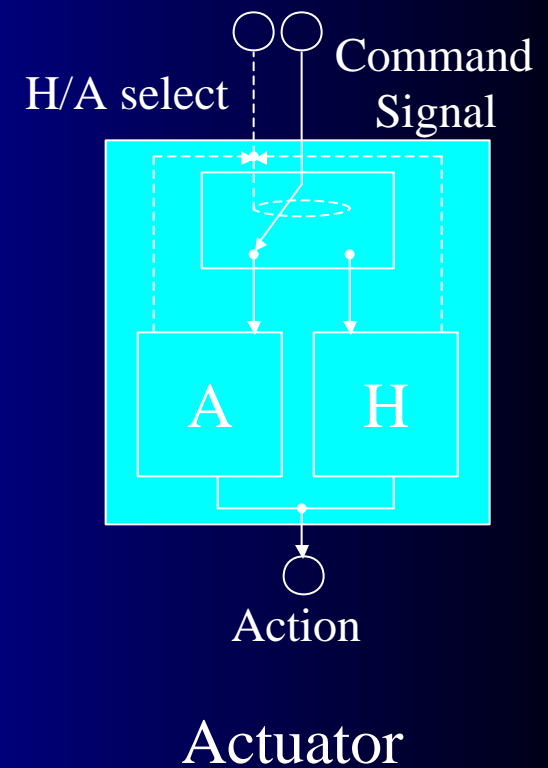
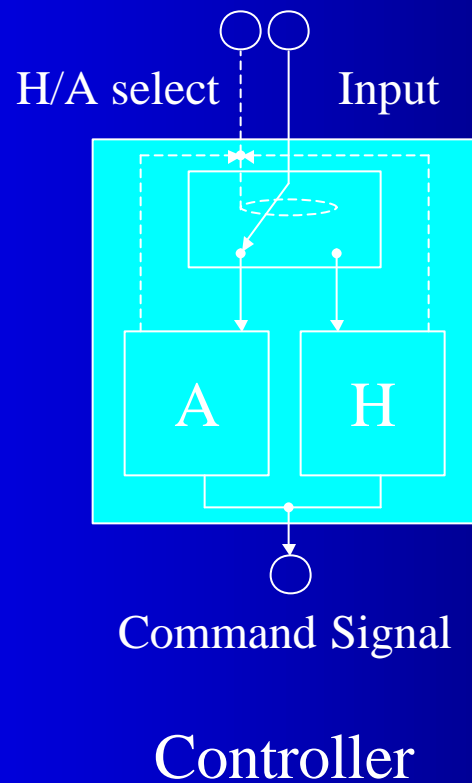
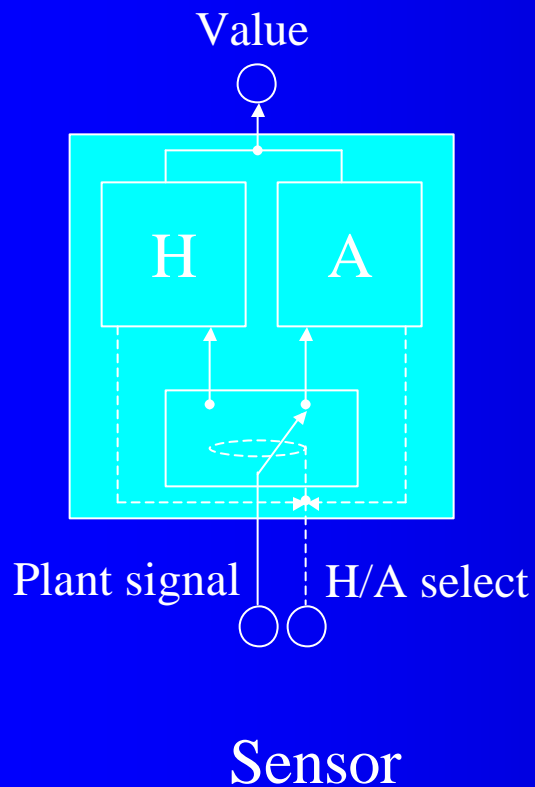


Our running example

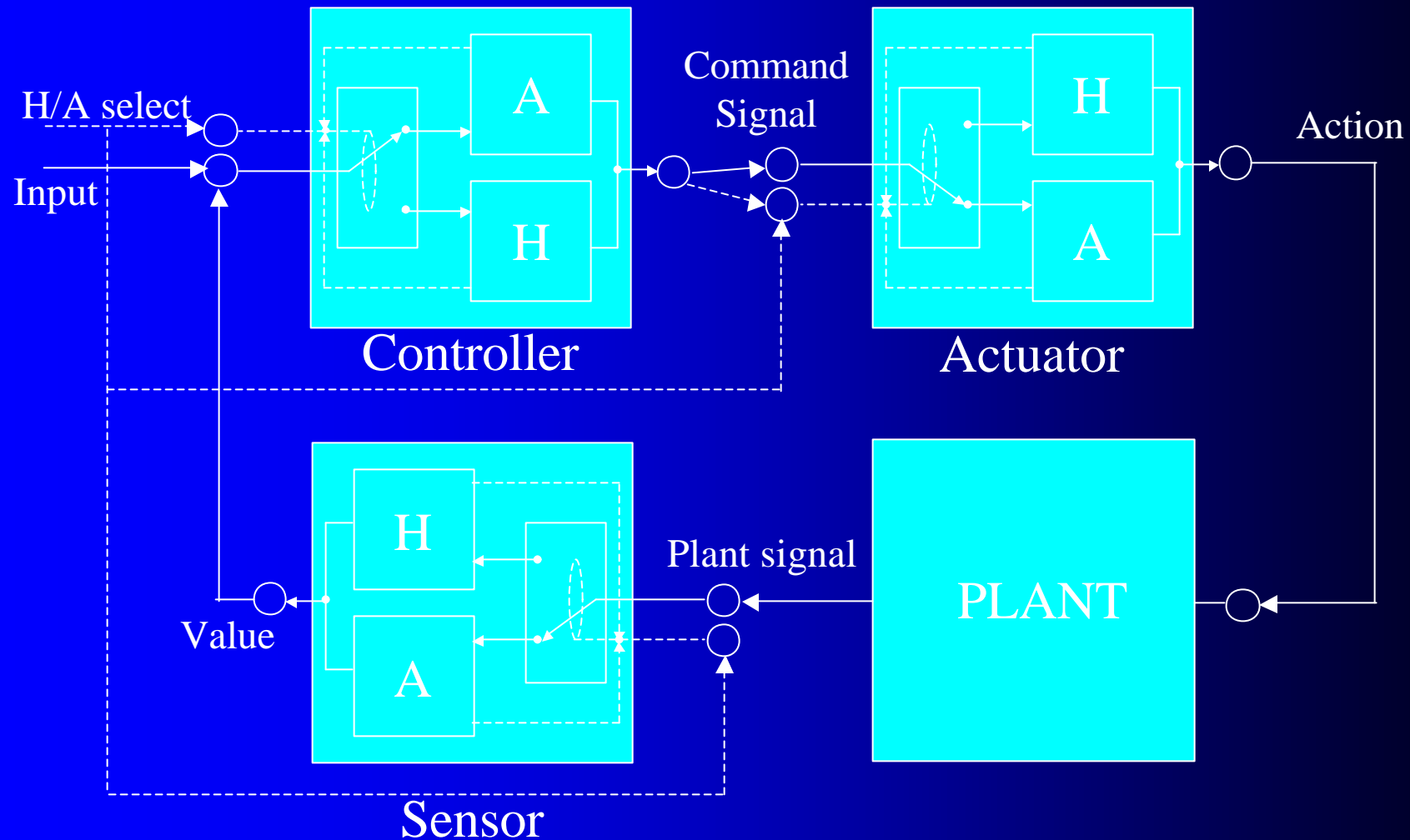
- Controller
 - human who decides whether pressure should be increased
 - computer that decides whether pressure should be increased
- Actuator
 - pump that human cranks
 - pump that motor activates
- Sensor
 - digital pressure sensor that computer reads
 - analog pressure gauge that human reads



AA control system components



A Simple AA Control System



Different autonomy modes



Adjusting Autonomy Example

- Control system believes motor is broken and is no longer responding to commands
 - Belief is either by internal processes or external command
- Autonomy is switched from autonomous (motor) to crank (human) for pump
- Autonomous control system still decides when pump needs to be activated
 - Human is only an actuator, not a controller

Example RAP Code

```
(define-rap increase-pressure ?maximum-pressure
  (context (equal actuator-autonomy-level 'A)
    (task1 '(turn-on-pump))
    (wait-for (pressure-at ?maximum-pressure))
    (task2 '(turn-off-pump)))
  (context (equal actuator-autonomy-level 'H)
    (task1 '(tell-user "Start cranking pump"))
    (wait-for (pressure-at ?maximum-pressure))
    (task2 '(tell-user "Stop cranking pump"))))
```

Labeling autonomy levels

- Difficult for human operator to know or set level of autonomy
- Design labels to help
 - tele-operated
 - supervisory
 - autonomous
 - ...
- Labels can be tied to specific tasks

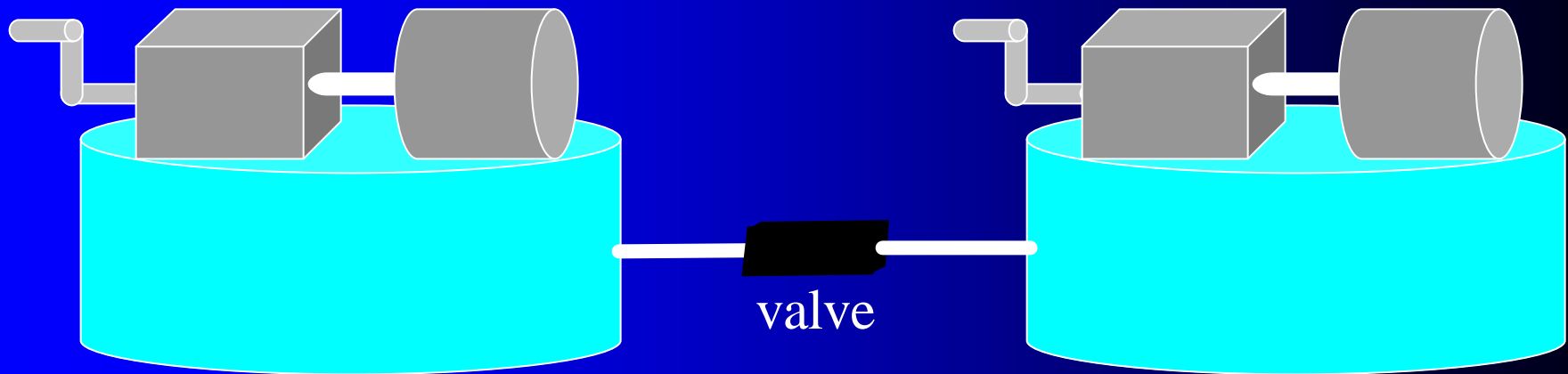
Communicating autonomy level

- Human operator needs to know the current state of the system, including autonomy level
- Must be in a form that is easy to understand “at a glance”
- Autonomy level of subsystems may be different
 - how to represent global autonomy level

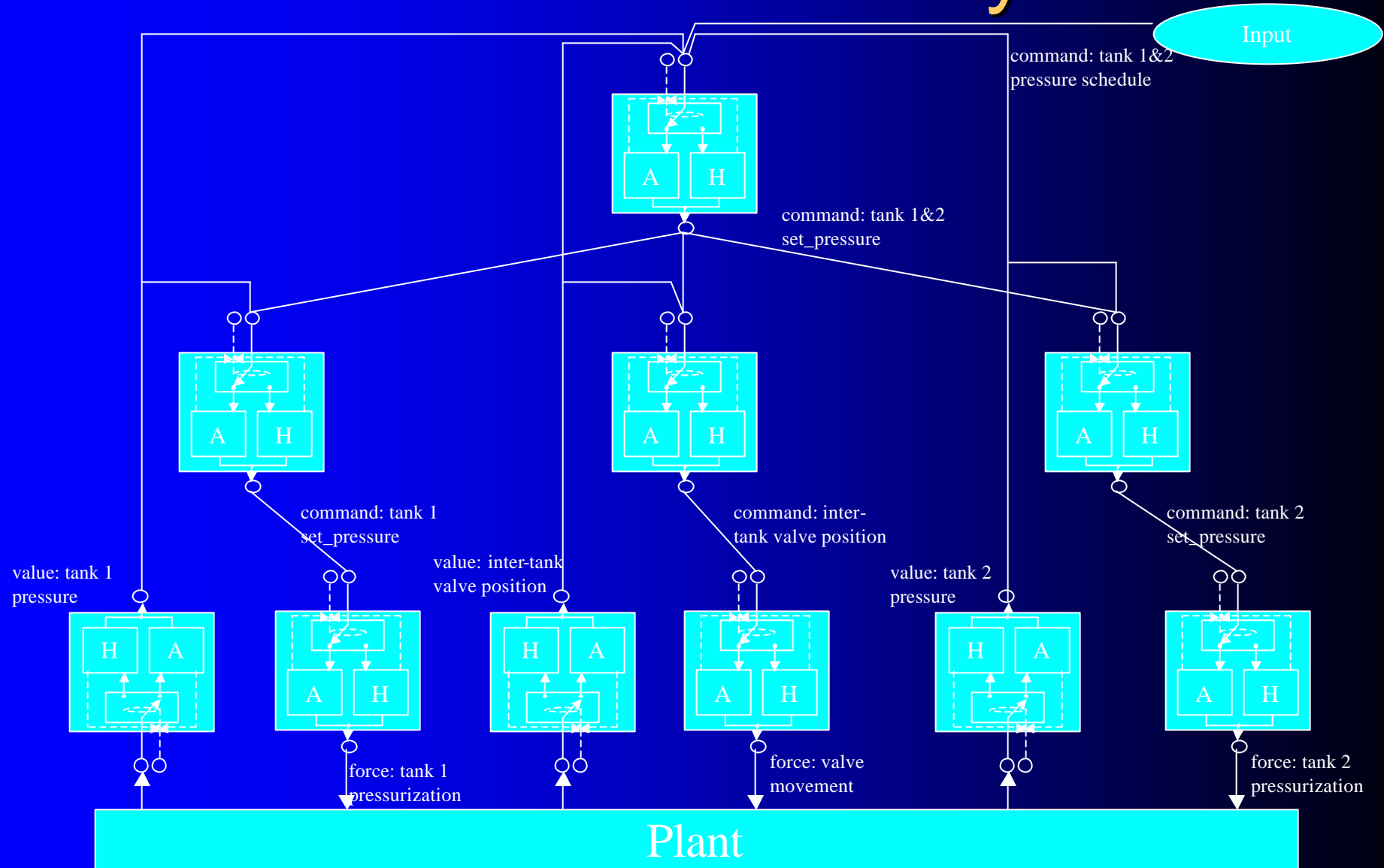
AA Hierarchies

- Multiple sensors, actuators and controllers make the combinatorics of adjustable autonomy nasty
- Arranging adjustable autonomy in a hierarchy can help alleviate these problems
 - setting level of autonomy at top of hierarchy and have it apply to all systems below
- Significant design decisions in arranging hierarchy
 - what level(s) can talk directly to sensors/actuators
 - is it a tree or a graph?

Expanding our example























A 2-tank AA control system



Autonomy Level Selection

Autonomy Level Selector

	A	H
Sensor #1		
Sensor #2		
Sensor #3		
Actuator #1		
Actuator #2		
Actuator #3		
Controller #1		
Controller #2		
Controller #3		
Controller #4		

For a system with:

s sensors

a actuators

c controllers

The possible number of
autonomy levels = $2^{(s+a+c)}$

In this case, $2^{10} = 1024$

Autonomy Level Considerations

- Not all autonomy levels valid, e.g., Actuator #2 must always be set to H and at least one sensor must always be set to H.
- It may not be permitted to directly switch from one valid autonomy level to another valid autonomy level, you may be required to transition to one or more intermediate autonomy levels. The number of possible transitions: $(n^2-n)/2$ where n = number of autonomy levels. When $n=1024$, the number of transitions is 523,776.

Autonomy Level Considerations cont.

- If a transition may be initiated only by a human (with the proper clearance level), or only by the autonomous system or either, then the number of possible transitions doubles to $n^2 - n$.
- Undesirable cycles may form, e.g., human selects an autonomy level and the autonomous system reselects the previous autonomy level.
- The set of valid autonomy levels and transitions may change over time, e.g., change of personnel shifts, equipment maintenance, or failure.

Design decisions

- What tasks can be done only by humans? Only by automation? By both?
 - Are there certain times or situations when a task should only be done by a human or automation?
- Who can set the level of autonomy for a task?
 - Can the level of autonomy change at any time or only under certain circumstances
- How will autonomy level(s) be set?
- How will autonomy level(s) be recognized?
- What will the system hierarchy look like?

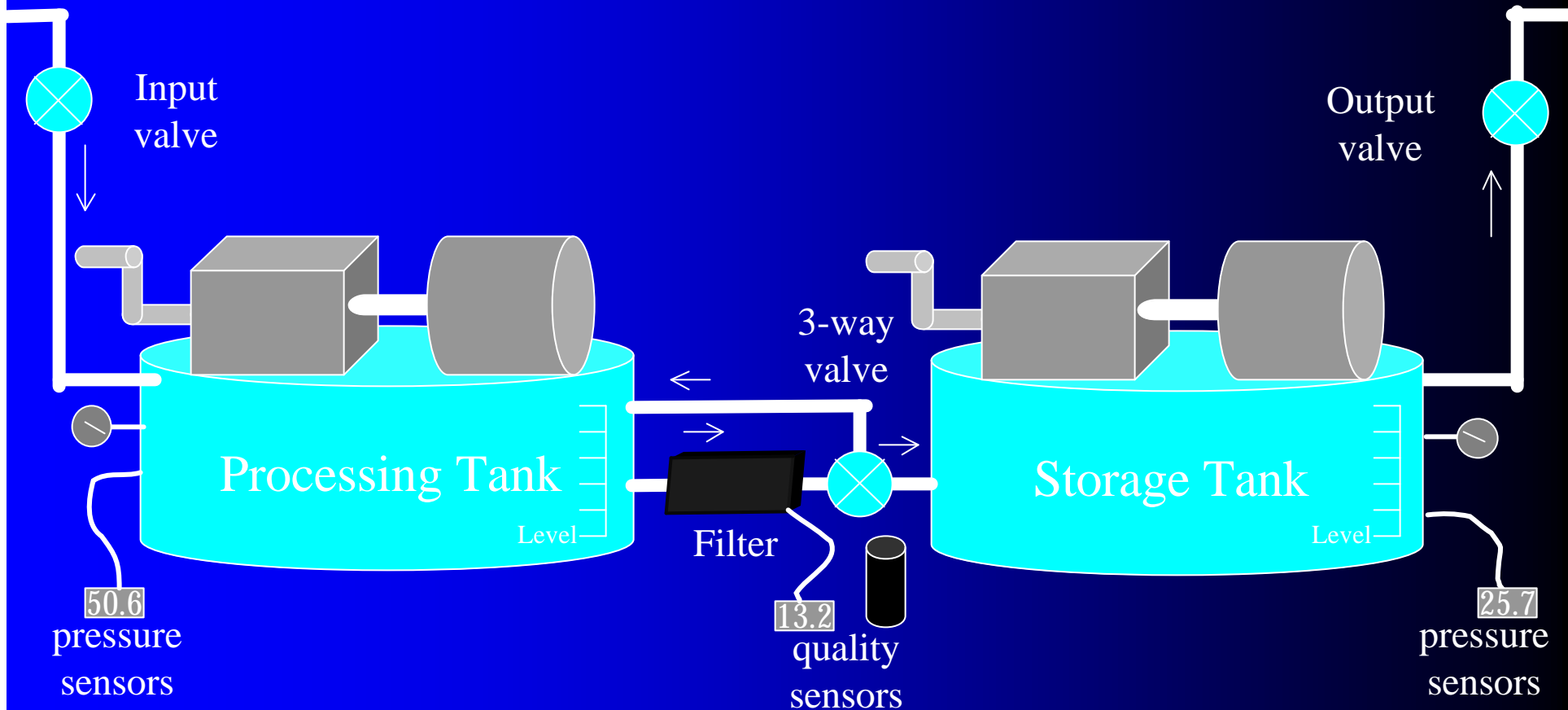
Requirements Outline

- Motivations
- Expanded Tank Example
- Accessibility of information
- Commandability
- Encode multiple methods to accomplish same task
- Knowledge of other available agents
- Changing responsibility: both accept and request

Motivations

- Adjustable autonomy places severe requirements on control systems
- Without a properly designed control system, adjustable autonomy can be ineffective and even dangerous
- Difficult to “retrofit” adjustable autonomy into existing autonomous or tele-operated control systems

Expanded Tank Example



Goal and Procedures for example

- Goal

- process product from processor tank to storage tank

- Procedures

- maintain higher pressure in processor tank than in storage tank
- if product quality is bad then recirculate
- if processor tank too full turn off input valve
- if storage tank empty close output valve

Accessibility of information

- Human (or other agent) who wants to adjust autonomy needs to know what the control system knows and what it is doing
- This includes information about
 - state
 - goals
 - tasks
 - models

State information

- State is a set of values that represent the current abstraction of the system (internal state) and its environment (external state)
- Human should be able to read and update:
 - internal states of control system
 - control system's perceived state of the world
 - from our example:

Internal State	External State
system setpoints	levels of tanks
autonomy level	quality of product
	valve positions

Models

- Models define set of possible states and their relationships.
 - typical ways to express models include rules, logical statements, equations, procedures, probability distributions.
- Models should be presented in a way that is easily understood by humans.
 - this includes allowing the user to predict futures states given the current state and a course of action as well as modifying the models that make such predictions.

Examples of Models

Model type	example																						
rule	if filter quality < 10.5 then close valve to storage tank																						
logical statement	pressure status = ($p_{t1} < 15$) and ($p_{t2} < 12$)																						
equation	inter-tank flow = $f(p_{t1} - p_{t2})$																						
procedure	while (quality sensor > 15) and ($p_{t1} > p_{t2}$) {set 3-way valve to open}																						
probability distribution	<p style="text-align: center;">Filter Condition Estimate</p> <table border="1"> <caption>Data points estimated from the Filter Condition Estimate graph</caption> <thead> <tr> <th>kiloliters of product</th> <th>probability of failure</th> </tr> </thead> <tbody> <tr><td>5</td><td>0</td></tr> <tr><td>10</td><td>5</td></tr> <tr><td>15</td><td>15</td></tr> <tr><td>20</td><td>35</td></tr> <tr><td>25</td><td>75</td></tr> <tr><td>30</td><td>90</td></tr> <tr><td>35</td><td>95</td></tr> <tr><td>40</td><td>98</td></tr> <tr><td>45</td><td>99</td></tr> <tr><td>50</td><td>100</td></tr> </tbody> </table>	kiloliters of product	probability of failure	5	0	10	5	15	15	20	35	25	75	30	90	35	95	40	98	45	99	50	100
kiloliters of product	probability of failure																						
5	0																						
10	5																						
15	15																						
20	35																						
25	75																						
30	90																						
35	95																						
40	98																						
45	99																						
50	100																						

Goal information

- Goal is a desired set of states.
- User needs to know the system's current goals and its progress in achieving those goals. System may need to explain nonlinearities, e.g., backtracking.
- For our example the main goal is to move water from the processing tank to the storage tank.
 - there may be sub-goals, e.g.,
 - recirculate product in processing tank
 - increase pressure in storage tank to 15

Task information

- Current tasks
 - task parameters
 - start time
 - estimated end time
- Finished tasks
 - finish time
 - success or failure
- Planned tasks
 - estimated start time

Commandability

- A system can be adjustably autonomous only if it can be commanded by an outside agent.
- Commanding can be divided into several types:
 - physical actuation, e.g., *turn pump 1 motor on*
 - goals, e.g., *maintain tank 1 pressure between 15 and 17 for 8 hours*
 - state, e.g., current product quality is 5
 - models, e.g., increase filter quality limit from 10.5 to 12.5 in the rule:
if filter quality < 10.5, then close valve to storage tank
 - task procedure execution, e.g., *execute procedure 7a*

Encode multiple methods to accomplish same task

- Adjustable autonomy only applies when there are multiple methods (paths) to accomplish system tasks.

Tank example: if the system only had one tank with a motorized pump and automated sensors and one tank with manual pump and sensors, you would still need to handle human-computer interactions but not adjustable autonomy.

Encode multiple methods to accomplish same task cont.

- The multiple task methods can be created manually or by using an automated planner/scheduler. The methods can be predetermined or created on demand.
 - automated planning of methods is helpful when there are a wide variety of possible ways to achieve the goals or there are many constraints that disqualify potential methods.

Tank example: The crew schedule may be such that crew members are only available for certain periods of times and each crew member may be only qualified to perform a subset of the possible manual tasks. Moreover, the electricity to run the motors may be highly constrained and vary over time. In such a case, an automated planner is useful to achieve goals.

Encode multiple methods to accomplish same task cont.

- For predetermined manually-created methods, agents should be provided succinct information regarding when/where each method is best applied. This information should include, but not be limited to:
 - probability of success
 - criteria for recognizing success/failure/no progress
 - resources available/required
- For planner-generated methods, agents should be provided heuristics to help the planner find and select plans for a wide variety of situations.

Knowledge of other available agents

- Minimum: (1) Human and (1) Autonomous system
- Capabilities
 - in order for an agent to delegate a task to another agent, it must know if that agent has the necessary capabilities.
 - one technique is for each agent to dynamically “publish” its capabilities to a registry.

Tank example: the system must know when a crew member is available to turn the pump and that a crew member can turn only one pump crank at a time at a maximum rate of x with a latency of y and a maximum duration of z .

Knowledge of other available agents cont.

- Commanding
 - includes task request, constraints (e.g., temporal, conflicts, preconditions)
 - grants or rescinds authority to do task, use resources, issue commands to other agents

***Tank example:** a crew member must be able to command the system not to turn on the storage pump motor for a period of time while the crew member performs maintenance on it. The system may ask if a crew member is available to crank the pump during this period and communicate the consequences if no pumping will be done.*

Knowledge of other available agents cont.

- Recognition of success/failure of these agents
 - success conditions, timeouts, resulting state

Tank example: The system may request a crew member to manually operate the pump and recognize when the desired pressure is obtained by a specified time or event.

- Prediction
 - system should anticipate need to change responsibility, whether to override or request human control

Tank example: An improved system would also recognize if the crew member is pumping at a rate that would enable the goal to be met. The system could then be proactive in making a correction, like switching to the electric motor, instead of waiting for the inevitable failure to occur.

Changing responsibility: both accept and request

- Probabilistic reasoning/latency/reactivity
 - under certain conditions, the user may want to select agent that minimizes risk, other times an agent that maximizes productivity

Tank example: There is uncertainty about the accuracy of the continuous product quality sensor attached to the filter. Under certain conditions, the system may request a manual check of the product quality and stop production until confirmed.

Changing responsibility: both accept and request cont.

- Safety
 - transition between responsible agents for goal/task
 - handoff should be planned to minimize risk when possible
 - system should verify with humans what their responsibilities are
 - system may support granting people different levels of command authority

***Tank example:** When the system requests a crew member to maintain a specified tank pressure, the system should request that the crew member acknowledge acceptance of the responsibility. Otherwise, the human may assume the pressure is being maintained by the system and the system assumes it is maintained by the human. In such a situation, the pressure is uncontrolled.*

Changing responsibility: both accept and request cont.

- Consistency

- the system should insure that constraints entered into the system are not violated regardless of the combination of methods being simultaneously executed by the system and users.

Tank example: There may be a constraint such that both pumps cannot be operated simultaneously when the difference between the tank pressures exceed a specified value.

Changing responsibility: both accept and request cont.

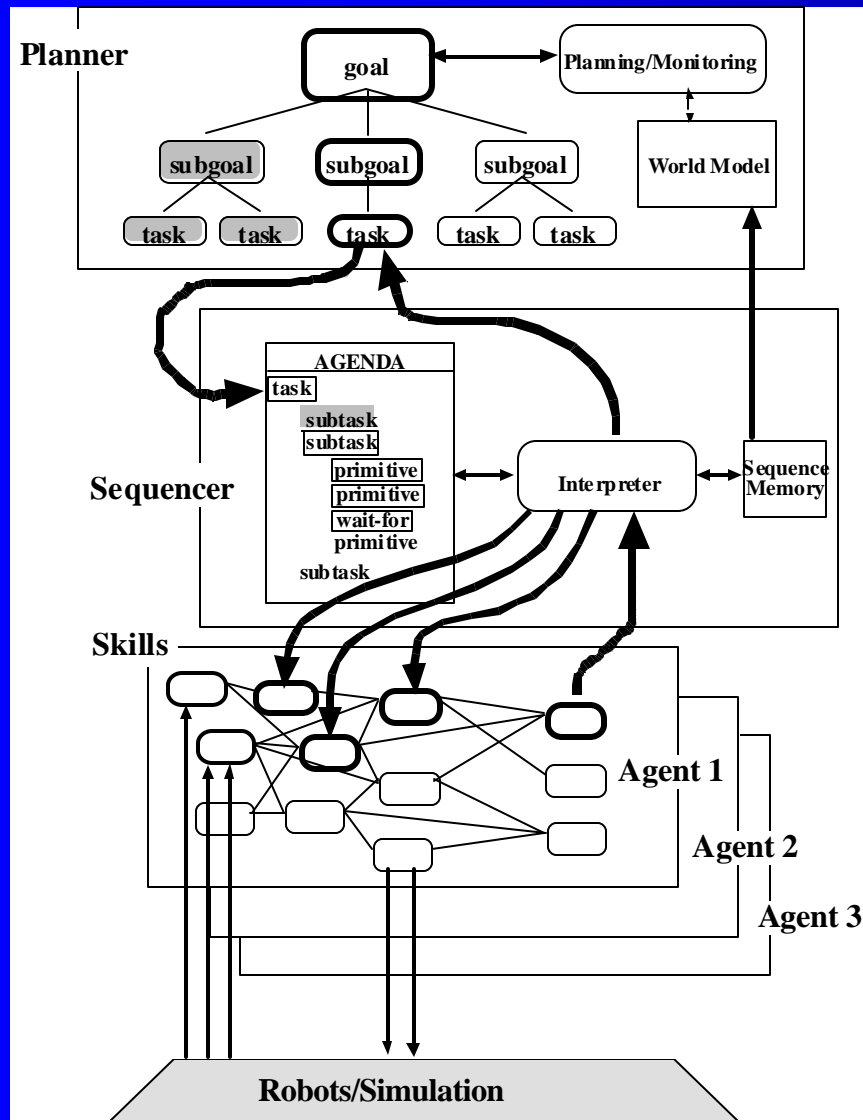
- Recognize unsafe transitions
 - system should warn human of potentially undesirable consequences of overriding a recommended autonomous behavior.
 - even when the system is not responsible for controlling certain sub-systems, it can advise users when constraints are being violated or prevent commands that would violate such constraints from being executed.

Tank example: The system should either warn or prevent the user from pressurizing the tank above the set limit or opening certain valves when it is in an over-pressurized state.

NASA HCA Applications Outline

- 3T Control Architecture
 - architecture description
 - Life Support Product Gas Transfer
 - Shuttle Remote Manipulator System Assistant
 - BIO-Plex
- Remote Agent Architecture
 - architecture description
 - Deep Space One spacecraft demonstration
 - Personal Satellite Assistant (under development)

3T Control Architecture



- **Planning**

- responsible for time and resource constraints
- Adversarial Planner [Elsaesser and Sanborn 1990]

- **Sequencing**

- conditional activation of skill sets
- Reactive Action Packages (RAPs) [Firby 1987] and Task Description Language (TDL) [Simmons and Apfelbaum, 1998]

- **Control**

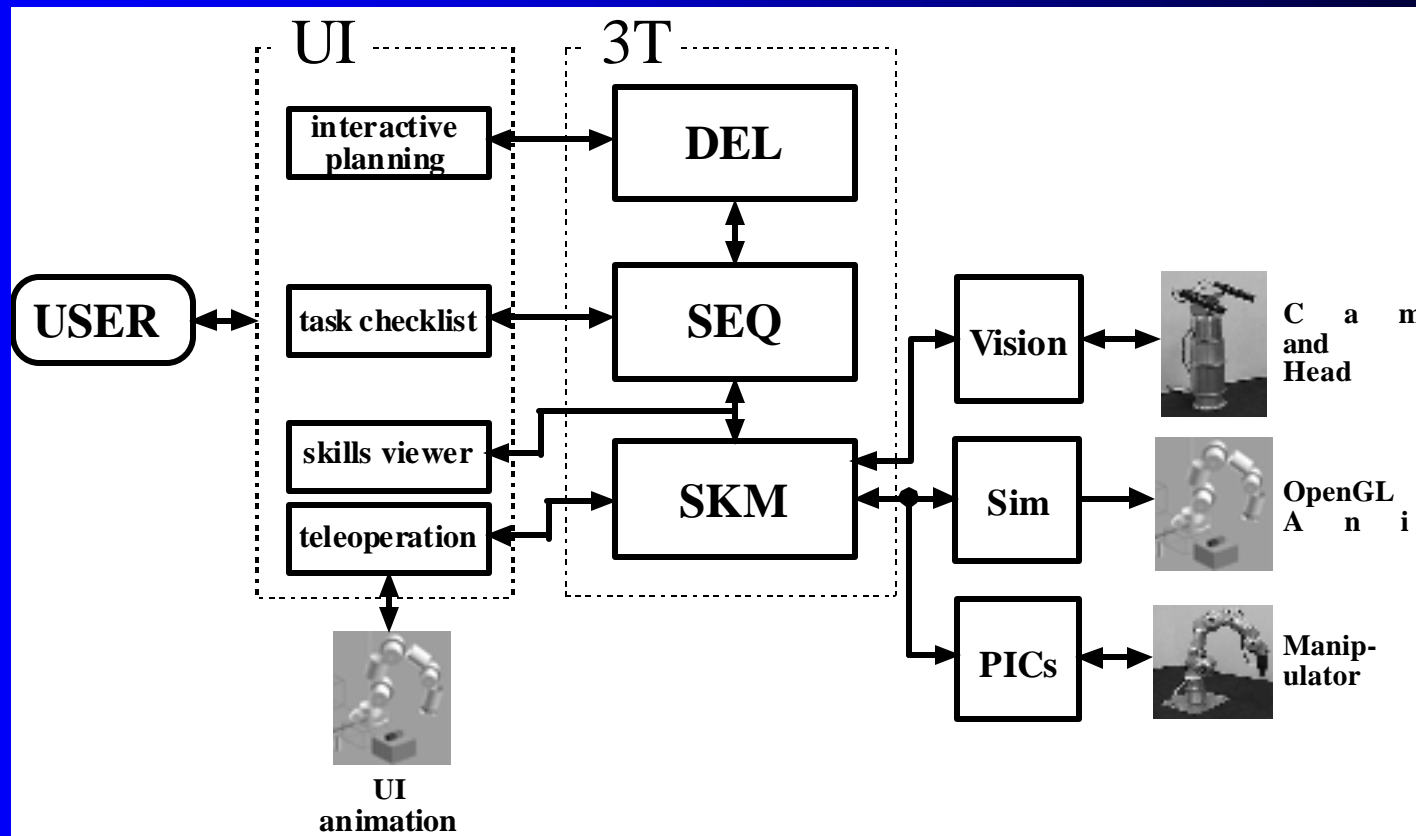
- skills provide reactive control of robot
- embedded in hardware (PC104 and VME) and executed in real-time in VxWorks
- Skill Manager Development Environment (SMDE)

6-DOF Arm w/ gripper testbed

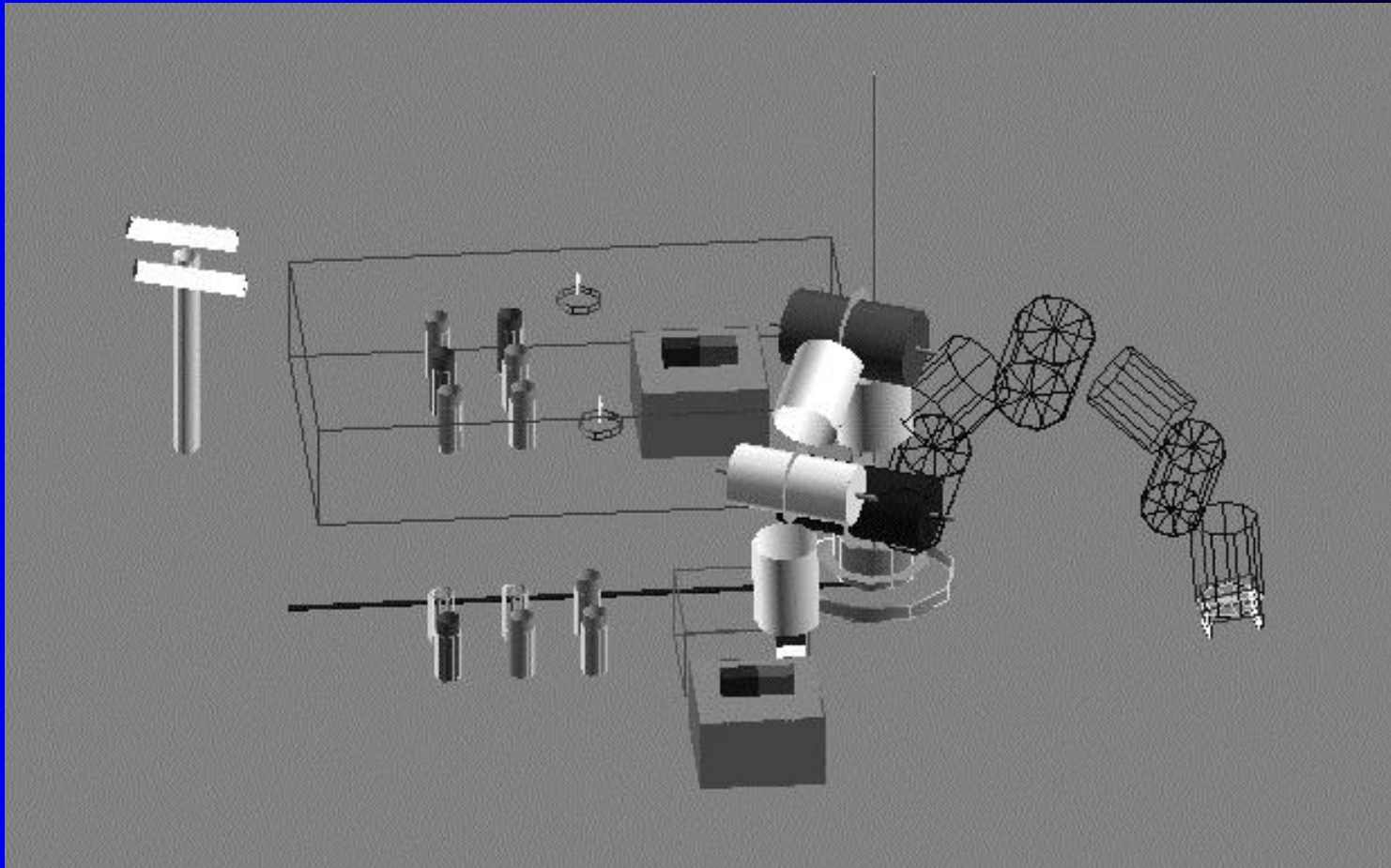


- Pan/tilt/verge head and color cameras
- Full dynamic simulation of manipulator, gripper, head and environment
- Computers for control, 3T/user interface and vision

User Interfaces

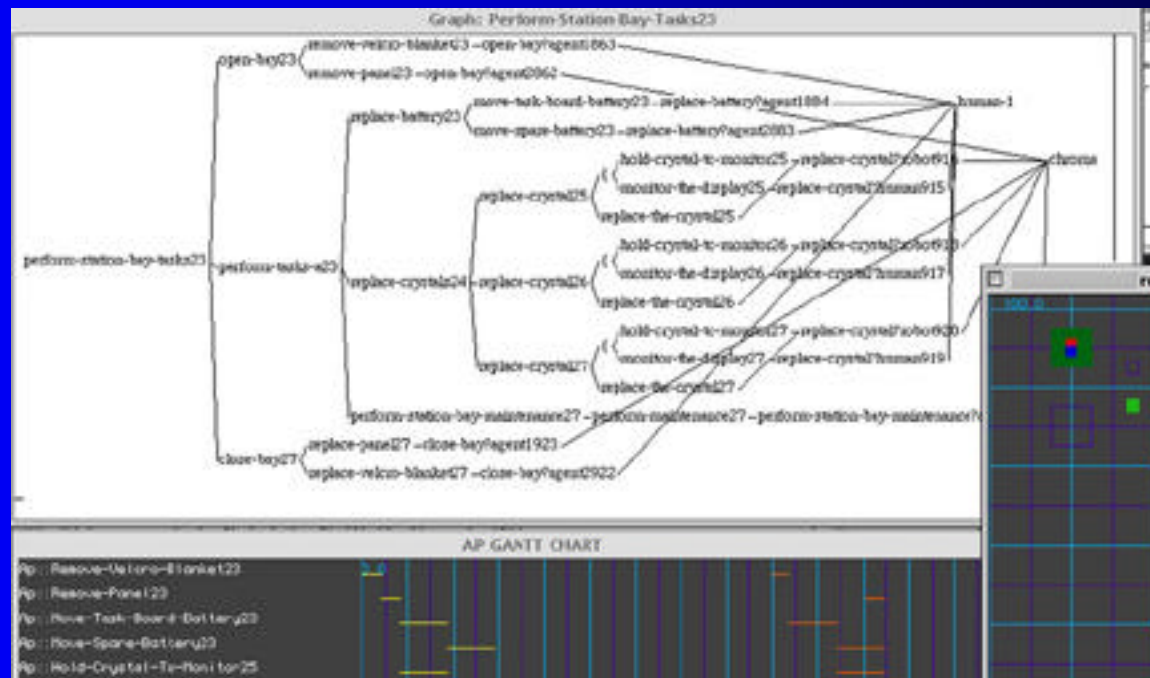


User Interface Animation



Planning

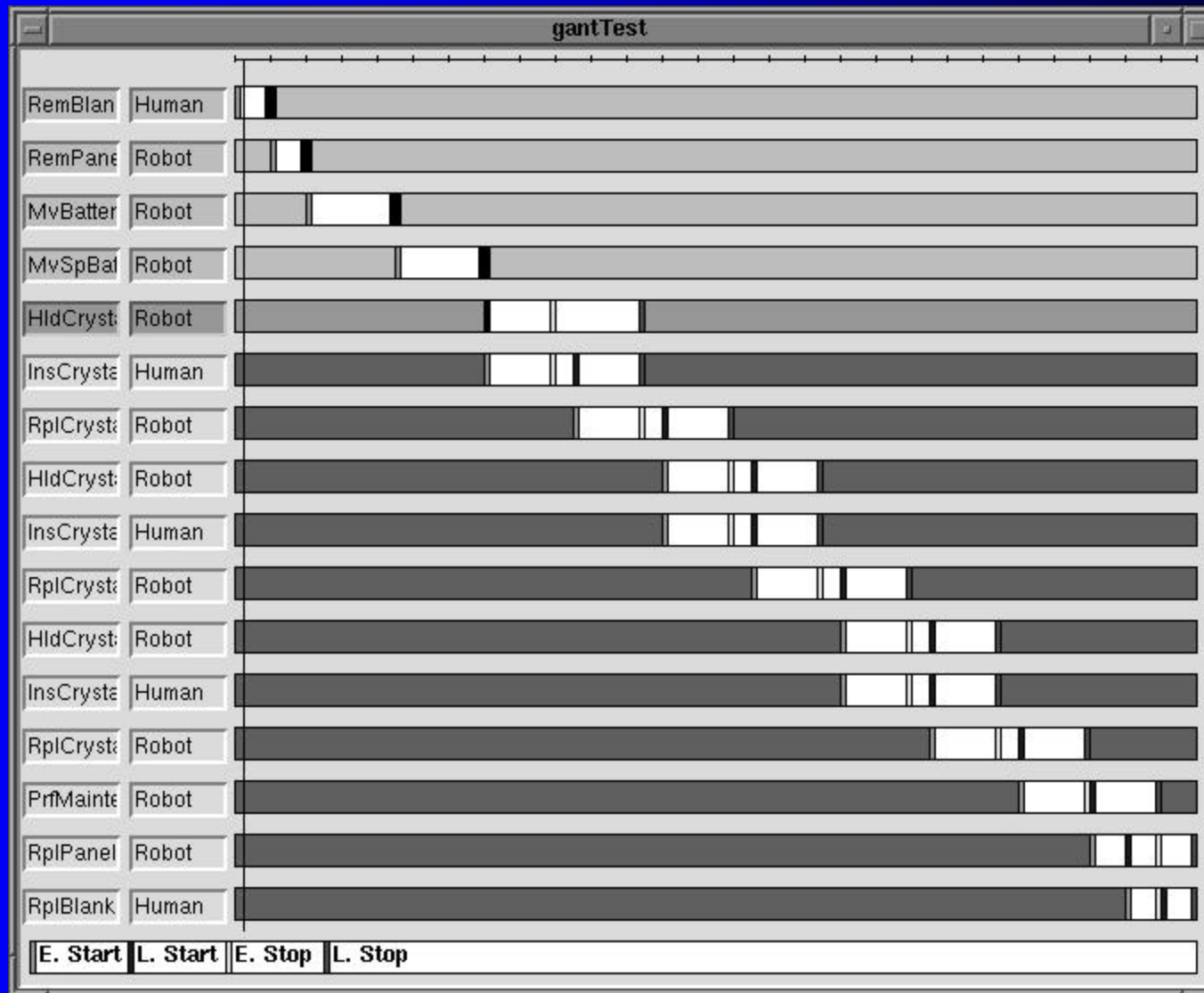
- Hierarchical decomposition
- Multi-agent planning



Plan Operators

```
(Operator replace-crystals
:purpose (state crystals replaced)
:agents (?robot ?human)
:constraints ((instance-of ?robot 'robot)
              (instance-of ?human 'human))
:preconditions ((state bay opened))
:plot (sequential
      (covers
       (monitor-crystals ?robot in-monitor)
       (display ?human monitored))
      (crystals ?robot are-replaced))
:effects ((state crystals replaced)))
```

Interacting with the planner



Sequencer

- Reactive Action Packages System (RAPS) from Firby with significant modifications
- Responsible for execution of task
- RAPS' succeed clauses match to planner's purpose clauses

Adjustable Autonomy RAP

```
(define-rap (arm-move ?arm ?place)
  (succeed (and (arm-at ?arm ?where)
                (= ?where ?place))))
(method robot-move
  (context (and (LOA arm-move ?arm ?place ?loa)
                (= ?loa autonomous))))
  (primitive
    (enable (:arm_move (:place . ?place))
      (wait-for (arm-move-done ?arm ?place ?result)
        :succeed (arm-move ?result))
      (disable :above)))
(method human-move
  (context (and (LOA arm-move ?arm ?place ?loa)
                (= ?loa tele-operate))))
  (primitive
    (tell-user "move arm to ?place")
    (wait-for (arm-move-done ?arm ?place ?result)
      :succeed (arm-move ?result))))))
```

Interacting with the sequencer

chklist

- PLAN STEP: REMOVE VELCRO BLANKET GOAL-SUCCEEDED at 1723786.0
- PLAN STEP: REMOVE PANEL GOAL-SUCCEEDED at 1723802.7
- PLAN STEP: MOVE TASK BOARD BATTERY

+ Grasp BATTERY Agent: ROBOT

Start: 1723807.1 Stop: 1723811.4

Manipulator State	CLOSED	CLOSED
Manipulator Contact	CONTACT	CONTACT

Move arm BATTERY REMOVED Agent: ROBOT

Start: 1723811.4 Stop:

Arm Position	BATTERY-IN-PLACE	BATTERY-REMOVED
Visual of Arm	BATTERY-IN-PLACE	BATTERY-REMOVED

Ungrasp BATTERY Agent: ROBOT

Start: Stop:

Manipulator State	CLOSED	OPEN
Manipulator Contact	CONTACT	NO-CONTACT

- PLAN STEP: MOVE SPARE BATTERY
- PLAN STEP: REPLACE PANEL
- PLAN STEP: REPLACE VELCRO BLANKET

chklist

- PLAN STEP: REMOVE VELCRO BLANKET GOAL-SUCCEEDED at 1723393.6
- PLAN STEP: REMOVE PANEL GOAL-SUCCEEDED at 1723410.9
- PLAN STEP: MOVE TASK BOARD BATTERY TIMEOUT at 1723432.5

+ Grasp BATTERY Agent: ROBOT

Start: 1723415.4 Stop: 1723419.5

Manipulator State	CLOSED	CLOSED
Manipulator Contact	CONTACT	CONTACT

X Move arm BATTERY REMOVED Agent: ROBOT

Start: 1723419.5 Stop: 1723425.6

Arm Position	BATTERY-IN-PLACE	BATTERY-REMOVED
Visual of Arm	BATTERY-IN-PLACE	BATTERY-REMOVED

X Ungrasp BATTERY Agent: ROBOT

Start: 1723425.6 Stop: 1723425.6

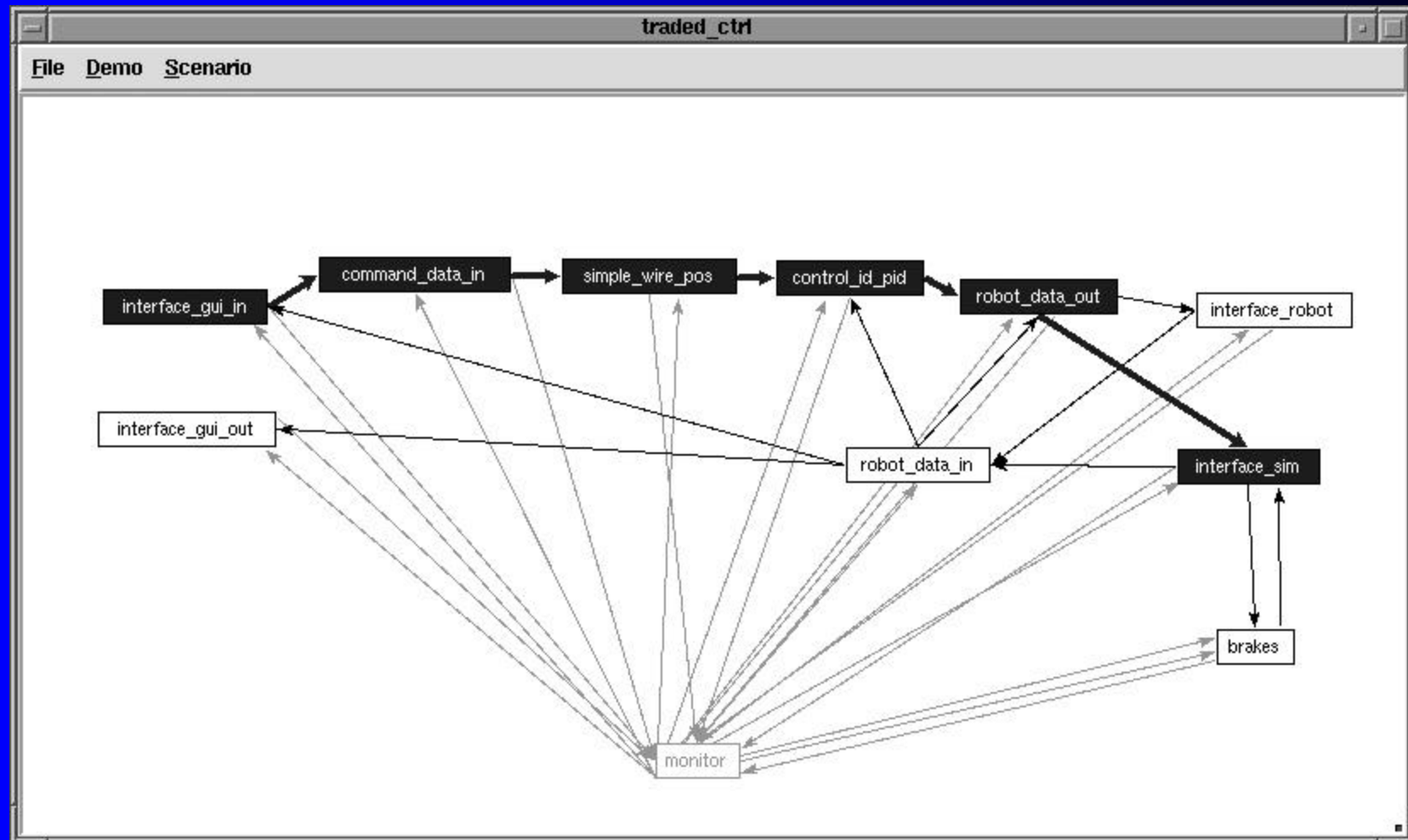
Manipulator State	CLOSED	OPEN
Manipulator Contact	CONTACT	NO-CONTACT

- PLAN STEP: MOVE SPARE BATTERY REMOVED at 1723432.5
- PLAN STEP: REPLACE PANEL REMOVED at 1723432.5
- PLAN STEP: REPLACE VELCRO BLANKET REMOVED at 1723432.5

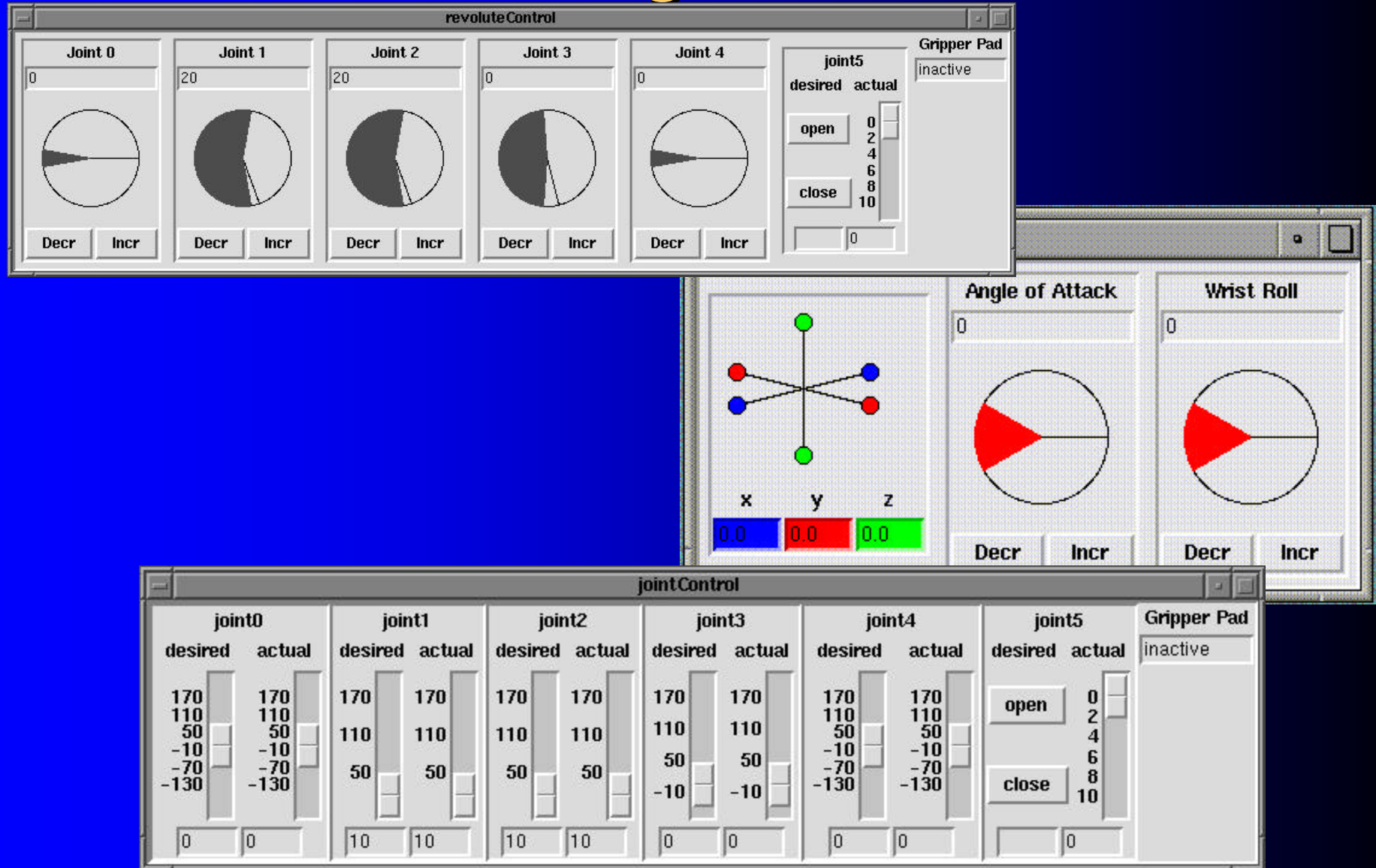
Skills

- Network of situated skills
 - skills exchange data
 - skills run at different frequencies
- Sequencer enables sets of skills to cause a specific robot behavior
- Skill manager handles:
 - communication with sequencer
 - scheduling of skills on CPU
 - movement of data between skills

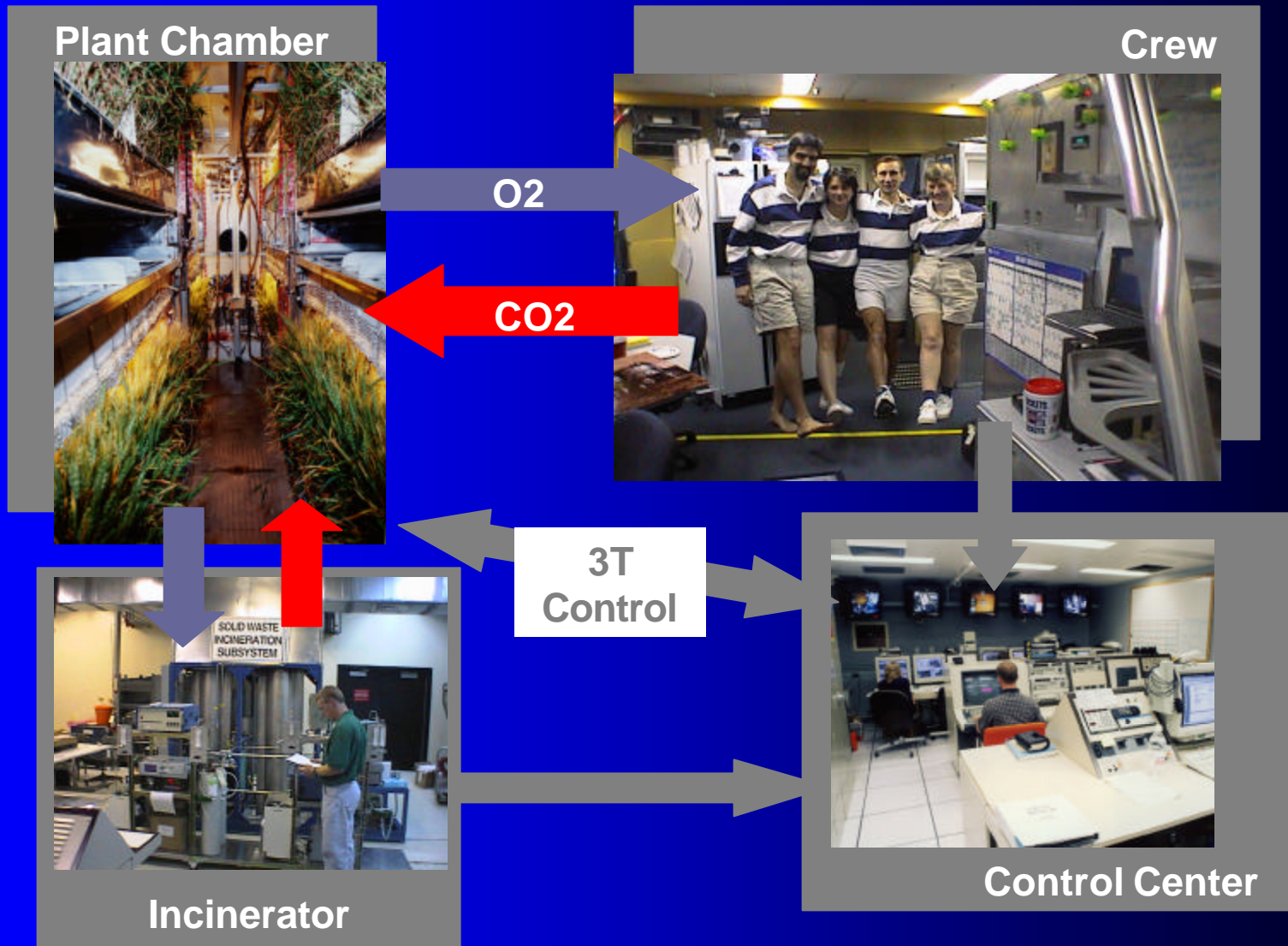
Skill network for task



Interacting with skills



Product Gas Transfer (PGT)

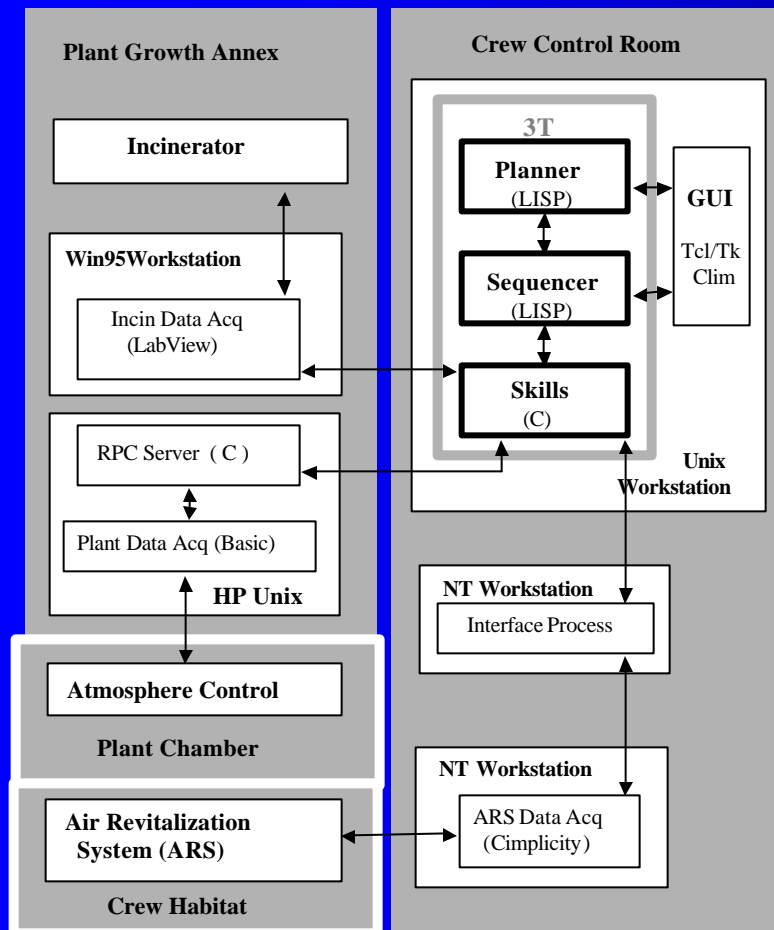


3T PGT Control Application

- Intelligent process control using 3T architecture
 - Resource planning: strategies for managing contended resources
 - Manage storage and use of oxygen for crew and waste incineration
 - Schedule airlock for crop germination/planting/harvesting and waste incineration
 - Reactive sequencing: tactics to control gas flow
 - Maintain O₂ & CO₂ concentrations in plant chamber
 - Maintain O₂ concentration in airlock during incineration
 - Configure the flow of oxygen and carbon dioxide
 - Detect caution & warning states and execute recovery procedures
 - Skill management: interface between hardware and software
 - Interface control software to life support instrumentation
 - Log data for analysis

Reference: Schreckenghost et al 1998

3T PGT Operations



- Operated reliably round-the-clock for 73 days (10/6-12/19)
 - Provided control, display, and data logging (15 sec data cycle)
- Typically ran without human supervision or intervention
- Required manual operations
 - Take gas analyzers offline for calibration (3 times weekly)
 - Reconfigure skill interfaces to receive data from incinerator
 - Transfer data files logged by 3T to centralized computer 3 times a week
 - Override autonomous control at planting and harvest (3 hours every 16-24 days)

RMS Assistant

- 3T as software framework for a procedure tracking system for the space shuttle Remote Manipulator System (RMS)
- Track expected steps of crew members and detect errors or malfunctions
- Autonomous operations tested in simulation

Reference: Bonasso, Kortenkamp and Whitney 1997b

RMS Assistant RAP

```
select-and-test-joint-p (agent duration timeout)
  method auto+
    context: (level-of-autonomy
              select-and-test-joint-p
              autonomous) AND
              (next-joint-for-test ?j) AND
              ((joint-tested ?j -) OR NOT
               (joint-tested ?j))

    ... etc.

  method auto-
    context: (level-of-autonomy
              select-and-test-joint-p
              autonomous) AND
              (next-joint-for-test ?j) AND
              (joint-tested ?j +)

    ... etc.

  method tele-operation
    context: (level-of-autonomy
              select-and-test-joint-p
              teleoperation)

  primitive
  enable:      tell-user "There are ~a more
                        joints to test."
                        (compute-joints-left)

  wait-for:    :correct-joint-response
                (:timeout 15) joint direction
                result

                :succeed (joint direction result)

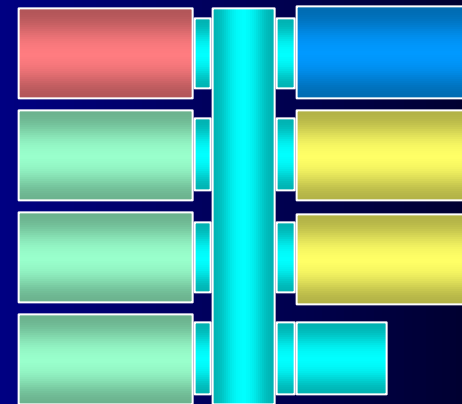
  disable     :above
```

BIO-Plex

- Project Goals

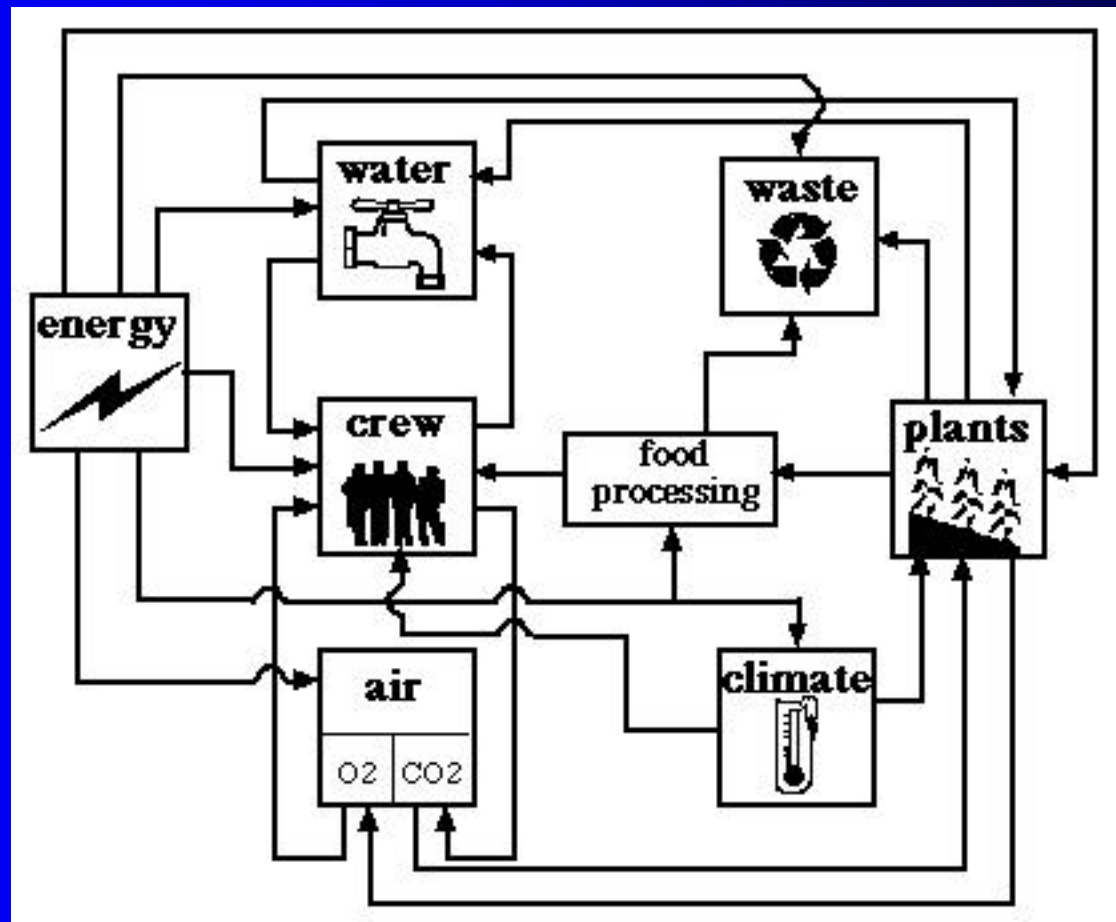
- Develop a high-fidelity test facility capable of evaluating large-scale bioregenerative planetary life support systems with human test crews for long durations
- Provide the Agency test bed capability for continued, extended bioregenerative life support technology development
- Serve as a focal point for other disciplines to conduct research and to develop supporting technologies, techniques, and procedures pertinent to future planetary missions via cooperative and collaborative experimentation and testing

- Habitation & human factors
- Medical
- Psychological
- Training
- Mission operations
- Automation & robotics
- Others

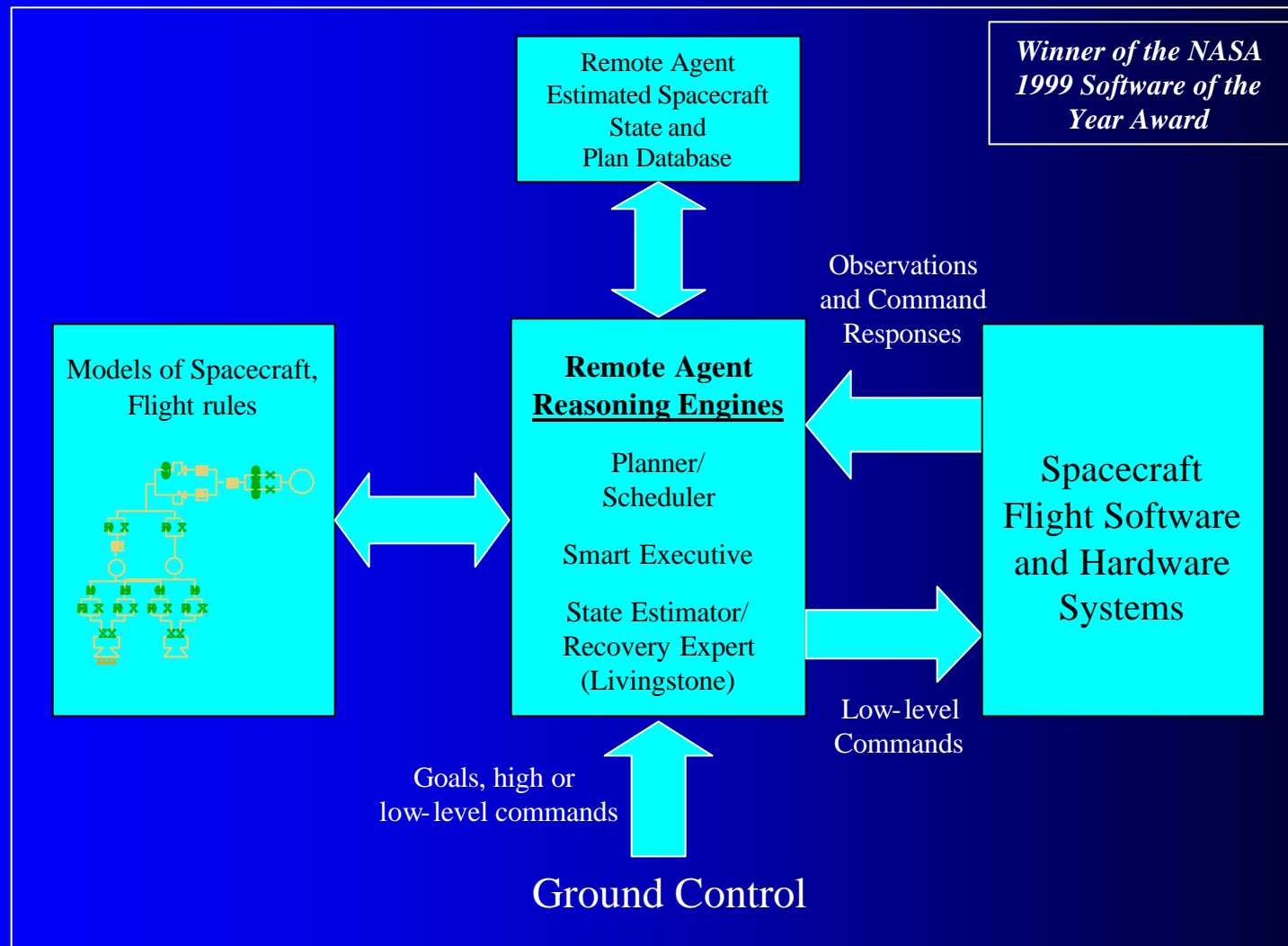


Reference: Kortenkamp, 2000

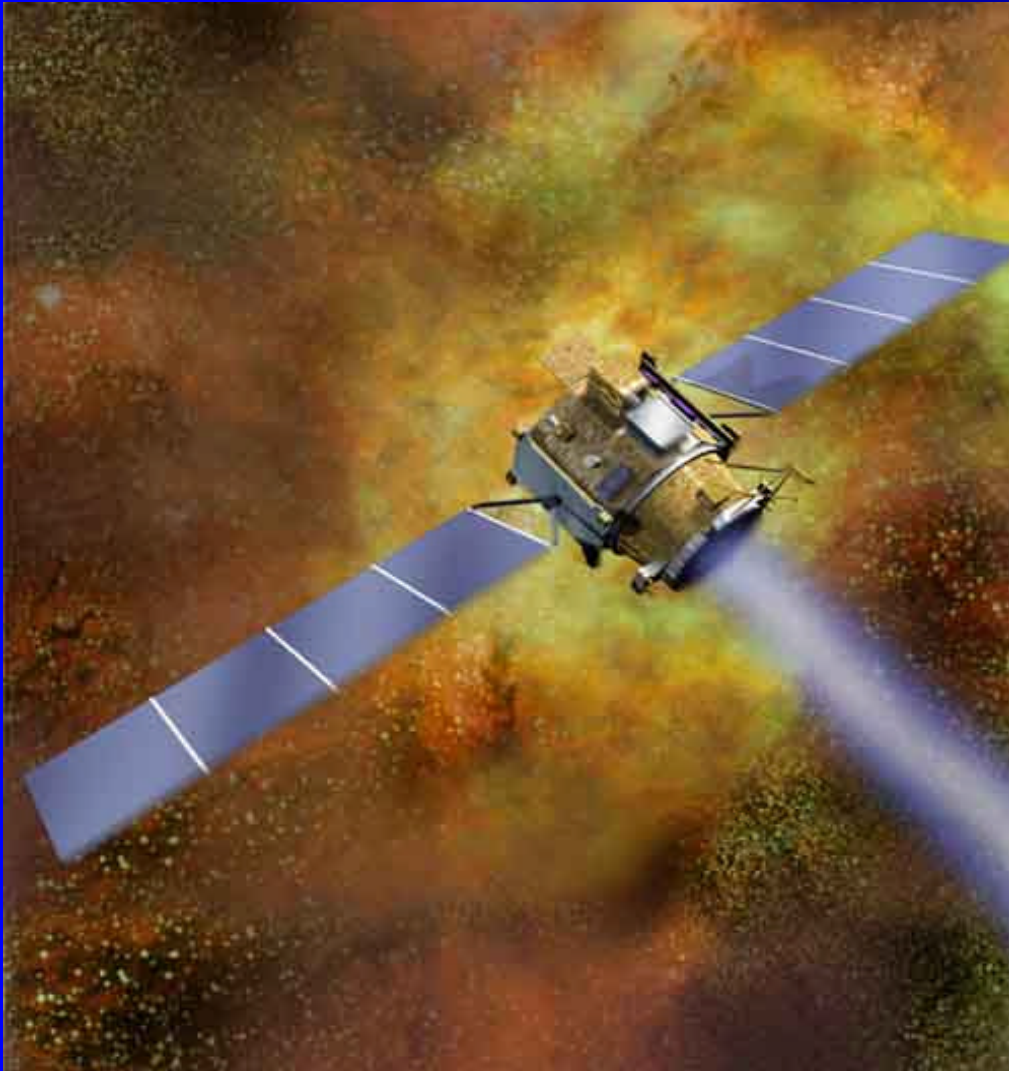
BIO-Plex subsystem interactions



Deep Space One Remote Agent



Deep Space One (DS1)



- Launched 10/98
- Remote Agent Experiment 5/99
- Other technologies flight-validated:
 - Ion Propulsion
 - SCARLET Solar Panels
 - Miniature Integrated Camera & Spectrometer
 - Autonomous Navigation
 - Beacon Monitor
- Current state:
thrusting toward comet
Borrelly for 9/2001 encounter

DS1 Remote Agent Architecture

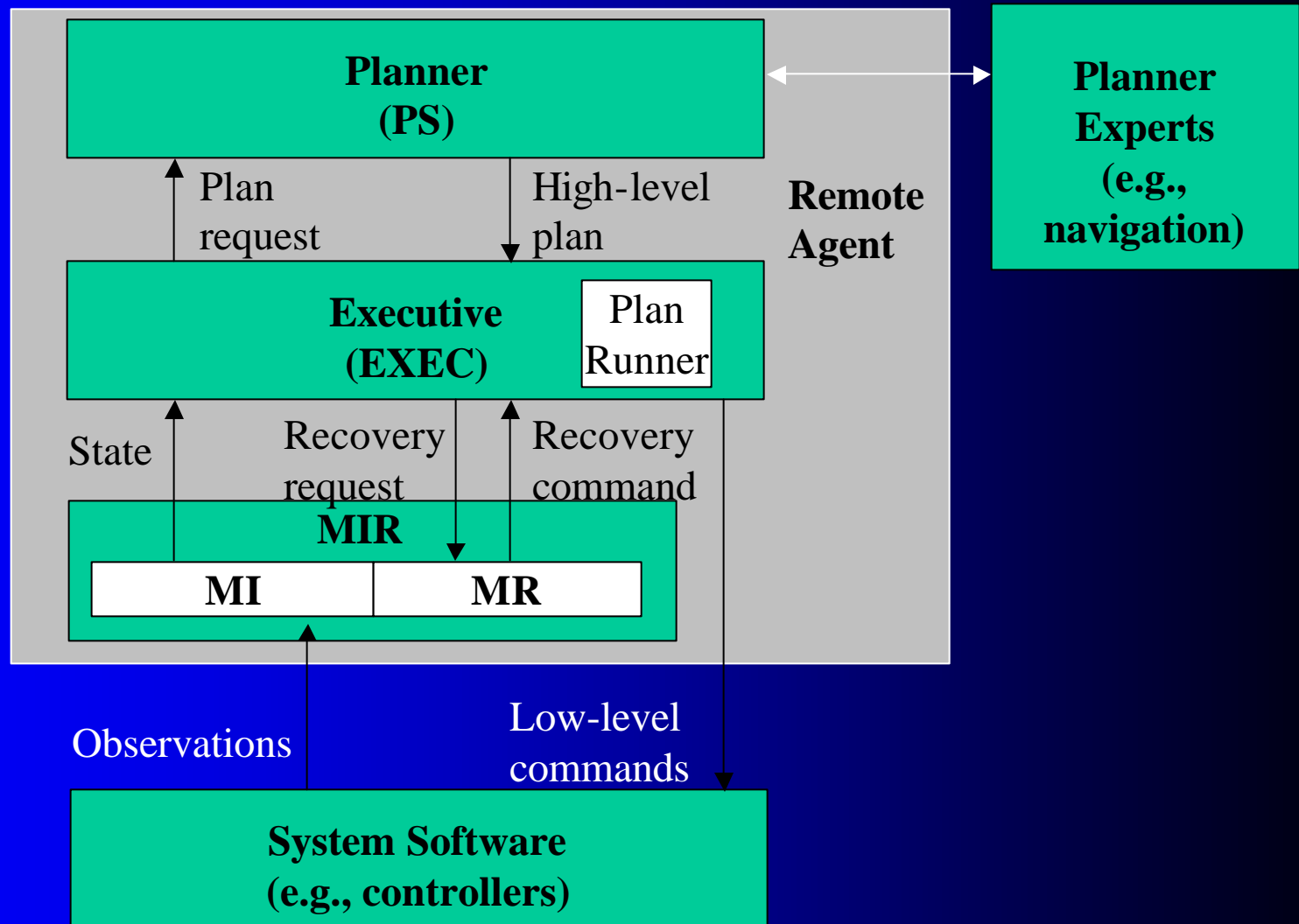
Abstraction Level

High-level
declarative
model

Medium-level
procedural
model

Low-level
declarative
model

Low-level
procedures



Domain Requirements

- Achieve diverse goals on real spacecraft
- High Reliability
 - single point failures
 - multiple sequential failures
- Tight resource constraints
 - resource contention
 - conflicting goals
- Hard-time deadlines
- Limited Observability
- Concurrent Activity

Approach

- Constraint-based planning and scheduling
 - supports goal achievement, resource constraints, deadlines, concurrency
- Robust multi-threaded execution
 - supports reliability, concurrency, deadlines
- Model-based fault diagnosis and reconfiguration
 - supports limited observability, reliability, concurrency
- Real-time control and monitoring

Diversity of Goals

- Final state goals
 - “Turn off the camera once you are done using it”
- Scheduled goals
 - “Communicate to Earth at pre-specified times”
- Periodic goals
 - “Take asteroid pictures for navigation every 2 days for 2 hours”
- Information-seeking goals
 - “Ask the on-board navigation system for the thrusting profile”
- Continuous accumulation goals
 - “Accumulate thrust with a 90% duty cycle”
- Default goals
 - “When you have nothing else to do, point HGA to Earth”

Diversity of Constraints

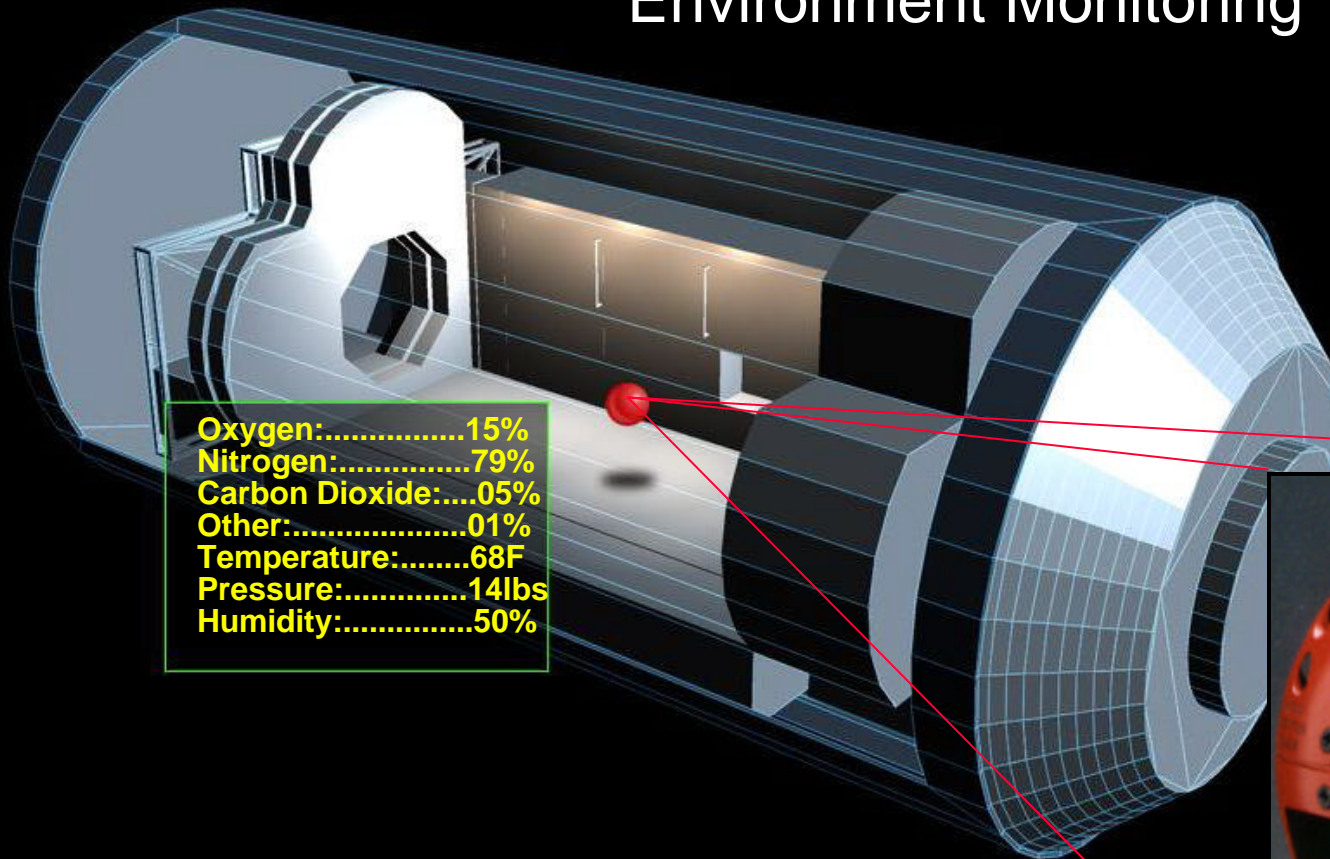
- State/action constraints
 - “To take a picture, the camera must be on.”
- Finite resources
 - power
- True parallelism
 - the ACS loops must work in parallel with the IPS controller
- Functional dependencies
 - “The duration of a turn depends on its source and destination.”
- Continuously varying parameters
 - amount of accumulated thrust
- Other software modules as specialized planners
 - on-board navigator

Deep Space One Remote Agent

- Levels of autonomy supported on DS1
(listed from least to most autonomous mode):
 - single low-level command execution
 - time-stamped command sequence execution
 - single goal achievement with auto-recovery
 - model-based state estimation & error detection
 - scripted plan with dynamic task decomposition
 - on-board back-to-back plan generation, execution, & plan recovery

Personal Satellite Assistant (PSA)

Environment Monitoring

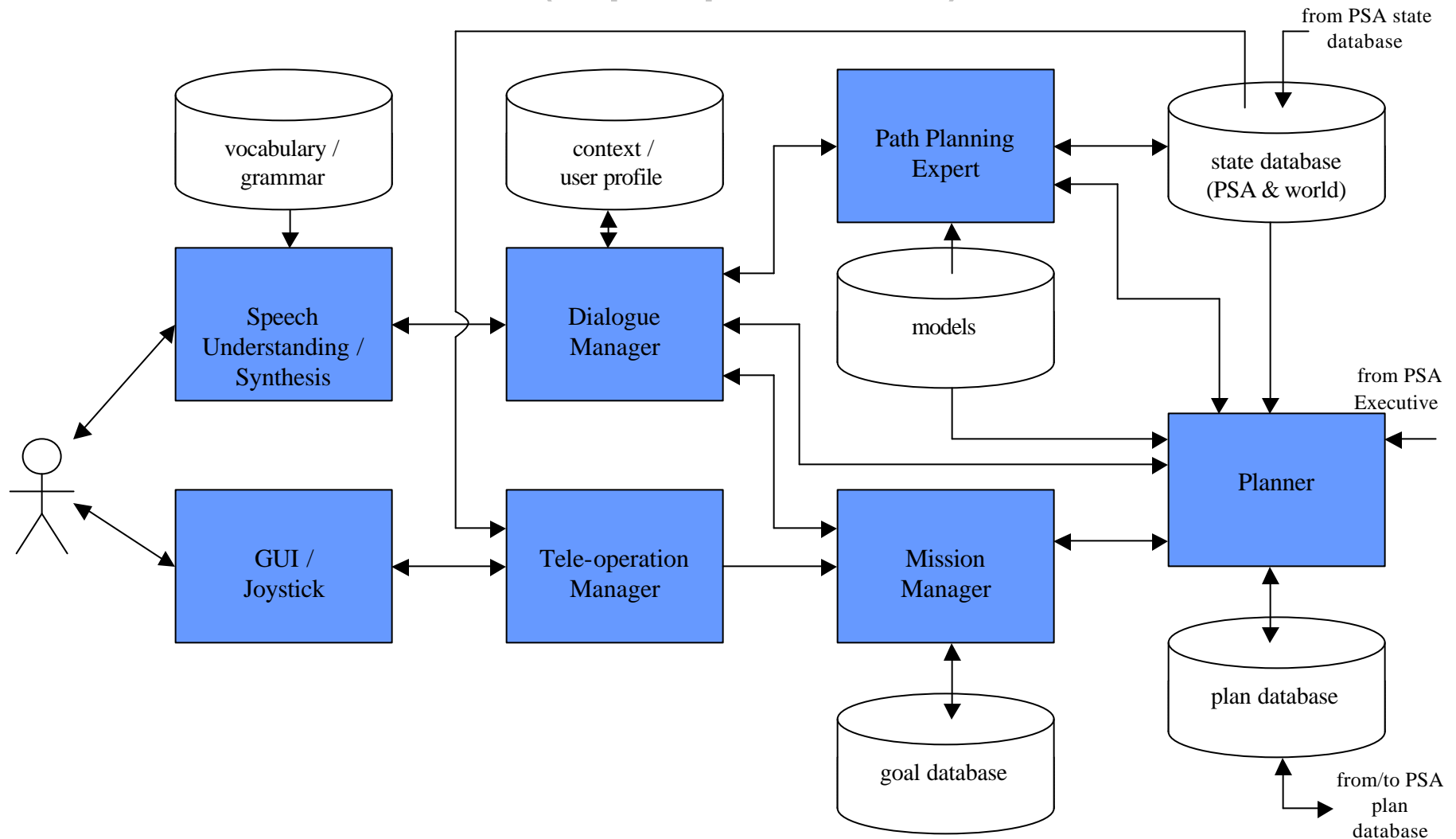


Drawing provided by Boris Rabin

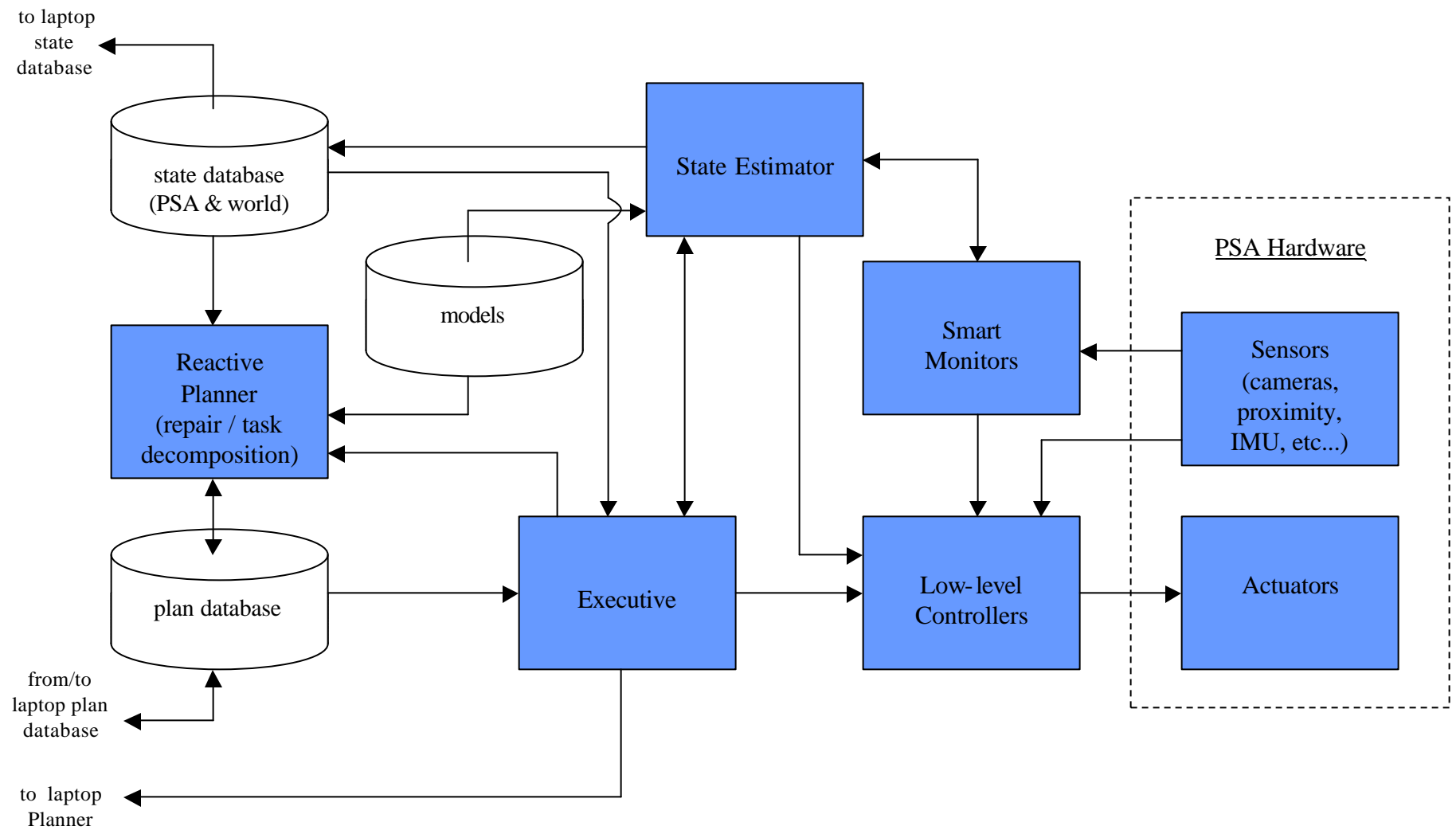
Space Station Module



Primary High-Level Control Modules (laptop server)



Primary Low-Level Control Modules (on-board PSA)



PSA Human-Centered Autonomy Requirements

- Humans dynamically modify plans during generation and execution
 - direct control can always be taken by human
- Humans dynamically act as sensors and actuators
- Humans dynamically modify domain models
- Humans communicate with system using limited natural-language context-sensitive grammar
- Autonomous system state, goals, models, and plans visible to humans
- System interacts with humans in its environment
- Dynamically modify plans as new goals are added, models change, human roles change, or plans fail

Conclusions

- Effective adjustable autonomy minimizes the necessity for human interactions but maximizes the capability for humans to interact at whatever level is most appropriate for any situation at any time.
- Adjustable autonomy must be designed in from the beginning -- assume pesky humans will always want to be meddling with the autonomous system!
- Often full autonomy is not possible (for technical, political or economic reasons) and adjustable autonomy is the only solution.
- By asking the right questions at design time (see the following checklist) adjustable autonomy can be safe and practical.

HCA Checklist

- What tasks can be done only by humans? Only by automation? By both?
 - are there certain times or situations when a task should only be done by a human or automation?
- Who can set the level of autonomy for a task?
 - can the level of autonomy change at any time or only under certain circumstances?
 - is the level of autonomy fixed at run-time or is it flexible?
- What are the timing issues with respect to a change in autonomy?

HCA Checklist cont.

- Arranging the hierarchy
 - can autonomy setting at one node apply to all descendants?
- What are possible autonomy level transitions?
What transitions are not permitted?
- Is information necessary to control the system available to the user or to other agents?
 - current state, tasks, goals
- Are there multiple ways to accomplish the same task? Are they selectable by the user? By the planner?

HCA Checklist cont.

- What parts of the system are commandable from outside?
 - by humans?
 - by other systems?
 - how are they commanded?
- How is success and failure of other agents recognized?
 - feedback?
 - observation?
 - timeout?

Citations

- Anderson, et al 1989, “Skill Acquisition and the LISP tutor,” Cognitive Science, 13, 467-506.
- Barber, et al 2000, “Dynamic Adaptive Autonomy in Multi-Agent Systems,” Journal of Experimental and Theoretical Artificial Intelligence.
- Bonasso, et al 1997a, “Experiences with an architecture for intelligence, reactive agents,” Journal of Experimental and Theoretical Artificial Intelligence, 9(1).
- Bonasso, et al 1997b, “Using a Robot Control Architecture to Automate Space Shuttle Operations,” 9th Conference on Innovative Applications of AI (IAAI-97).
- Burstein and McDermott, 1996, “Issues in the Development of Human-Computer, Mixed-Initiative Planning,” in Cognitive Technology: In Search of a Humane Interface, edited by B. Gorayska and J. L. May, Elsevier Science.
- Canamero, et al 1994, “Case-Based Plan Recognition in Dynamic Domains.” In Planning and Learning: On to Real Applications. Gil and Veloso (eds), AAAI Technical Report FS-94-01.
- Chambers and Nagel, 1985, “Pilots of the Future: Human or Computer?”, Communications of the ACM, 28(11), 1187-1199.
- Clancey, et al 1998, “Brahms: Simulating Practice for Work Systems Design,” International Journal of Human-Computer Studies, 49, 831-865.

Citations cont.

- Craik, 1947, “Theory of the Human Operator in Control Systems,” *British Journal of Psychology*, 38(1): 142-148.
- Decker and Lesser, 1995, “Coordination Assistance for Mixed Human and Computational Agent Systems,” Univ. of Mass. Comp. Science Tech. Report 95-31.
- Dorais, et al 1998, “Adjustable Autonomy for Human-Centered Autonomous Systems on Mars,” Mars Society Conference.
- Durfee, 1999, “Practically Coordinating,” *AI Magazine*, 20(1):99-116.
- Elsaesser and Sanborn, 1990, “An Architecture for Adversarial Planning,” *IEEE Transactions on Systems, Man, and Cybernetics*, v. 20, no. 1, pp. 186-194
- Ferguson, et al 1996, “TRAINS-95: Towards a Mixed-Initiative Planning Assistant,” *Proceedings Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*.
- Firby, 1987, “An investigation into reactive planning in complex domains,” *Proceedings of the National Conference on Artificial Intelligence*.
- Gertz, et al 1993, “A Human-Machine Interface for Reconfigurable Sensor-based Control Systems,” *Proceedings of the AIAA Conference on Space Programs and Technologies*.
- Gould, 1988, “How to Design Usable Systems,” In *Handbook of Human-Computer Interaction*, Helander (ed), North-Holland.

Citations cont.

- Grefenstette, et al 1990, “Learning sequential decision rules using simulation models and competition,” Machine Learning Vol. 5, No. 4, pp.355-381, Kluwer Academic Publishers, (NCARAI Report: AIC-90-010).
- Hayati and Venkataraman, 1989, “Design and Implementation of a Robot Control System with Traded and Shared Control Capability,” IEEE International Conference on Robotics and Automation: 1310-1315.
- Holland, 1992, *Adaptation in Natural and Artificial Systems*, MIT Press. 2nd edition. First edition 1975.
- Horvitz, et al 1998, “The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users.” Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, July 1998.
- Huber, et al 1994, “The Automated Mapping of Plans for Plan Recognition,” Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence.
- Jonsson, et al 2000, “Planning in interplanetary space: theory and practice,” Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems
- Jorgensen, 1997, “Direct Adaptive Aircraft Control Using Neural Networks,” NASA TM 47136.
- Kantz and Allen, 1986, “Generalized Plan Recognition,” Proceedings of the 5th National Conference on Artificial Intelligence, pp. 32-37.

Citations cont.

- Kortenkamp, 2000, “Real-time autonomous control of space habitats,” In Working Notes AAAI 2000 Spring Symposium on Real-Time Autonomous Control.
- Kortenkamp, et al 2000, “Adjustable Control Autonomy for Manned Space Flight,” IEEE Aerospace Conference.
- Kuipers, 1994, *Qualitative Reasoning*, MIT Press, Cambridge MA.
- Kuo, 1991, *Automatic Control Systems*, Prentice Hall, Englewood Cliffs, New Jersey.
- Lee, 1993, “Intelligent Sensing and Control for Advanced Teleoperation,” IEEE Control Systems Magazine, 13(3):19-28.
- Lesh, et al 1999, “Using Plan Recognition in Human-Computer Collaboration,” Proceedings of the Seventh Int. Conf. on User Modeling, Banff, Canada.
- Mitchell, 1997, *Machine Learning*, McGraw-Hill, Boston.
- Muscettola, et al 1998 “Remote Agent: To boldly go where no AI system has gone before,” *Artificial Intelligence*, 103(1), 5-47.
- Musliner and Krebsbach, 1999, “Adjustable Autonomy in Procedural Control for Refineries,” Working Notes of the AAAI Spring Symposium on Agents with Adjustable Autonomy.
- Musliner, et al 1995, “CIRCA: A Cooperative, Intelligent, Real-time Control Architecture,” *IEEE Transactions on Systems, Man and Cybernetics*, 23(6), 1561-1574.

Citations cont.

- Myers, 1996, “Advisable Planning Systems,” Advanced Planning Technology, Tate (ed), AAAI Press.
- Pollack and Horty, 1999, “There’s More to Life than Making Plans: Plan Management in Dynamic, Multi-Agent Environments,” AI Magazine, 20(4), 71-84.
- Roth, et al 1997, “Paradigms for Intelligent Interface Design,” The Handbook of HCI, edited by Helander, 2nd edition.
- Samuel, 1959, “Some studies in machine learning using the game of checkers,” IBM J. Res. Dev. 3:210-29.
- Schreckenghost, et al 1998, “Intelligent Control of Life Support Systems for Space Habitats,” Proceedings of the Conference on Innovative Applications of Artificial Intelligence.
- Schreckenghost and Malin, 1991, “Human-Computer Interaction with Intelligent Fault Management Systems during Space Operations. Volume I: Design Recommendations and Issues, Volume II: Case Study Results,” MITRE Technical Report, MTR-91W00146.
- Sheridan, 1989, “Telerobotics,” Automatica, 25(4), 487-507.
- Sheridan, 1992, *Telerobotics, Automation, and Human Supervisory Control*, The MIT Press, Cambridge MA.

Citations cont.

- Simmons and Apfelbaum, 1998, “A Task Description Language for Robot Control,” Proceedings Conference on Intelligent Robotics and Systems.
- Stein, 1988, “Understanding Why Things Go Wrong: Towards a Theory of Explanation and Plan Recognition,” AAAI Workshop on Plan Recognition.
- Tambe, et al 1999, “Building Agent Teams Using an Explicit Teamwork Model and Learning,” Artificial Intelligence 110(2): 215-239.
- Thurman, et al 1997, “An Architecture to Support Incremental Automation of Complex Systems,” Proceedings of the 1997 IEEE International Conferences on Systems, Man and Cybernetics.
- Weiss, 2000 (editor), *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge MA.
- Williams, 1996, “A Model-based Approach to Reactive Self-configuring Systems,” Proceedings 1996 National Conference on Artificial Intelligence (AAAI-96).